Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Kos

Postavitev in upravljanje računalniškega oblaka na platformi FIWARE

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Proučite področje računalništva v oblaku, analizirajte namestitvene modele, storitvene modele in opišite oblačno infrastrukturo. Podrobno analizirajte projekt FIWARE in posamezne gradnike. Opišite koncept modularnih gradnikov ter upravljanje z virtualnimi stroji in identitetami. Vzpostavite lasten zasebni oblak z uporabo FIWARE in opišite arhitekturo, okolje in postopek namestitve. Prikažite delovanje in način migracije obstoječih strežnikov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jan Kos sem avtor diplomskega dela z naslovom:

Postavitev in upravljanje računalniškega oblaka na platformi FIWARE

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Matjaža Branka Juriča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 12. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju dr. Matjažu Branku Juriču za strokovno pomoč in vodenje pri izdelavi diplomskega dela. Posebej se zahvaljujem svoji družini in punci, ki so me ves čas podpirali, me spodbujali in mi stali ob strani.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Računalništvo v oblaku **3**

2.1 Definicija 3

2.2 Namestitveni modeli 4

2.2.1 Javni oblak 5

2.2.2 Skupnostni oblak 5

2.2.3 Zasebni oblak 6

2.2.4 Hibridni oblak 7

2.3 Storitveni modeli 7

2.3.1 Infrastruktura kot storitev 8

2.3.2 Platforma kot storitev 9

2.3.3 Programska oprema kot storitev 10

2.4 Oblačna infrastruktura 11

2.4.1 Virtualizacija strežnikov 11

2.4.2 Virtualizacija shrambe 13

2.4.3 Virtualizacija omrežij 16

2.4.4 Virtualizacija omrežnih storitev 19

3 Projekt FIWARE **21**
3.1 Kaj je FIWARE? 21
3.1.1 Platforma FIWARE 22
3.1.2 FIWARE Lab 23
3.1.3 FIWARE Ops 23
3.1.4 FIWARE Accelerate 24
3.1.5 FIWARE Mundus 24
3.2 Koncept modularnih osnovnih gradnikov 25
3.2.1 Definicija 25
3.2.2 Razvoj 27
3.2.3 Objava v katalogu FIWARE 28
3.3 Upravljanje z identitetami - KeyRock 29
3.3.1 Keystone 29
3.3.2 Horizon 30
3.4 Upravljanje z virtualnimi stroji - DCRM 31
3.4.1 Glance 32
3.4.2 Nova 32
3.4.3 Cinder 34
3.4.4 Neutron 36
3.5 Samopostrežni portal - Cloud Portal 42

4 Vzpostavitev zasebnega oblaka FIWARE **45**
4.1 Arhitektura 46
4.2 Priprava okolja 48
4.3 Namestitev in konfiguracija komponent 49
4.3.1 Podporne storitve 49
4.3.2 Upravljanje z identitetami 50
4.3.3 Shranjevanje slik 51
4.3.4 Računska storitev 52
4.3.5 Omrežna storitev 53
4.3.6 Bločna shramba 54
4.3.7 Samopostrežni portal 55

4.4 Prikaz delovanja 57
4.4.1 Priprava slik 57
4.4.2 Izstavitev virov 60
4.4.3 Migracija obstoječih strežnikov 65

5 Zaključek **71**

Literatura **73**

Slike

2.1 Shema hibridnega oblaka. 6
2.2 Storitveni modeli računalništva v oblaku [1]. 8
2.3 Primerjava hipervizorjev različnih tipov in vsebovalnikov. . . . 12
2.4 Primerjava shrambe DAS, NAS in SAN. 14
2.5 Nutanix NDFS [2]. 15
2.6 Zgradba okvirja Ethernet pri uporabi oznak 802.1Q [3]. 16
2.7 Komunikacija preko navideznih lokalnih omrežij. 17
2.8 Komunikacija preko navideznih prekrivnih omrežij. 18

3.1 Katalog FIWARE. 28
3.2 Procesni tok Keystone za izvedbo generične zahteve [4]. 30
3.3 Bločna shramba z gonilnikom LVM [5]. 35
3.4 Arhitektura vtičnika ML2 [6]. 37
3.5 Arhitektura stikala Open vSwitch [7]. 39
3.6 Open vSwitch na računskem vozlišču in omrežnem vozlišču. . . 40
3.7 Samopostrežni portal FIWARE Cloud Portal. 42

4.1 Načrt arhitekture. 47
4.2 Dodajanje storitve za upravljanje identitet v katalog storitev. . 50
4.3 Digitalno potrdilo pri dostopu do portala. 56
4.4 Vnaprej definirani tipi instanc. 57
4.5 Uvoz slike navideznega diska Ubuntu LTS 14.04. 58
4.6 Ročno pripenjanje bločne enote in slike ISO z gonilniki. 59
4.7 Dodajanje gonilnikov VirtIO pri namestitvi Windows 8.1. . . . 60

4.8 Vmesnik samopostrežnega portala. 61
4.9 Uvoz javnega ključa SSH za dostop do instanc. 61
4.10 Določanje privzetega prehoda navideznemu usmerjevalniku. . . 62
4.11 Izbira javnega omrežja. 62
4.12 Interna omrežja. 62
4.13 Dodajanje internega omrežja. 63
4.14 Dodajanje vmesnika na usmerjevalniku. 63
4.15 Izbira slike. 64
4.16 Izbira velikosti in imena instance. 64
4.17 Izbira ključa SSH in varnostnih skupin. 64
4.18 Pripenjanje plavajočega naslova IP. 65
4.19 Obstoječi domenski strežnik Windows Server 2012 R2. 66
4.20 Dodajanje gonilnikov z orodjem pnputil. 67
4.21 Postopek migracije navideznega diska in zagon instance. . . . 68
4.22 Preseljen strežnik v samopostrežnem portalu. 68
4.23 Preseljen domenski strežnik. 69

Seznam uporabljenih kratic

kratica	angleško	slovensko
IaaS	Infrastructure as a Service	infrastruktura kot storitev
PaaS	Platform as a Service	platforma kot storitev
SaaS	Software as a Service	programska oprema kot storitev
NIST	National Institute of Standards and Technology	Nacionalni inštitut za standarde in tehnologijo
SLA	Service-Level Agreement	dogovor na ravni storitve
VM	Virtual Machine	navidezni stroj
DAS	Direct-attached storage	direktno priključena shramba podatkov
NAS	Network-attached storage	omrežno dostopna shramba podatkov
SAN	Storage area network	omrežje podatkovnih shramb
VLAN	Virtual Local Area Network	navidezno lokalno omrežje
TRILL	Transparent Interconnection of Lots of Links	transparentna medsebojna povezanost več povezav
NFV	Network functions virtualization	virtualizacija omrežnih funkcij
FI-PPP	Future Internet Public-Private Partnership	javno-zasebno partnerstvo za internet prihodnosti

kratica	**angleško**	**slovensko**
GE	Generic Enabler	osnovni gradnik platforme FIWARE
SE	Specific Enabler	namenski gradnik platforme FIWARE
API	Application Programming Interface	aplikacijski programski vmesnik
IoT	Internet of Things	internet stvari
I2ND	Interface to Networks and Devices	vmesniki za omrežja in naprave
REST	Representational State Transfer	slog spletne storitve
SQL	Structured Query Language	strukturirani povpraševalni jezik
LDAP	Lightweight Directory Access Protocol	protokol za dostop in upravljanje z imeniškimi storitvami
HTTP	HyperText Transfer Protocol	hipertekstovni prenosni protokol
AMQP	Advanced Message Queuing Protocol	napredni protokol za sporočilno vrsto
SSH	Secure Shell	varen protokol za upravljanje računalnika na daljavo
LVM	Logical Volume Management	upravljalec logičnih diskov
GRE	Generic Routing Encapsulation	protokol za enkapsulacijo
IP	Internet Protcol	internetni protokol
DHCP	Dynamic Host Configuration Protocol	omrežni protokol za dinamično nastavitev gostitelja

kratica	**angleško**	**slovensko**
DNS	Domain Name System	sistem domenskih imen
NTP	Network Time Protocol	protokol omrežnega časa
MTU	Maximum Transmission Unit	maksimalna velikost okvirja
NFS	Network File System	omrežni datotečni sistem
VNC	Virtual Network Computing	protokol za prikaz oddaljenega virtualnega namizja
SSL	Secure Sockets Layer	sloj varnih vtičnic
TLS	Transport Layer Security	zaščita transportne plasti

Povzetek

Vse več podjetij se odloča za selitev svojih informacijskih storitev v bodisi javni, zasebni ali hibridni oblak. Zmanjševanje stroškov, prilagodljivost in hitrejša izvedba so zgolj nekatere od prednosti, vendar ob tem ne smemo pozabiti na nevarnosti, ki jih tak model prinaša. Ena izmed ključnih ovir pri selitvi storitev v oblak je zaklepanje uporabnikov znotraj ponudnika, saj različni ponudniki ponujajo lastne nestandardizirane formate podatkov in aplikacij. Pomemben faktor pri uporabi lastniških rešitev predstavlja tudi visok strošek licenčnin. V diplomskem delu se zato lotimo odprte tehnološke platforme FIWARE, ki predstavlja odprto alternativo uveljavljenim lastniškim rešitvam za postavitev oblačne infrastrukture. Glavni cilj je vzpostavitev in evalvacija zasebnega oblaka na odprti platformi z namenom nudenja infrastrukture kot storitve. Najprej definiramo koncepte računalništva v oblaku in preučimo tehnologije, ki te koncepte omogočajo, pri čemer se osredotočimo na nudenje infrastrukture kot storitve. Ogledamo si projekt FIWARE in se podrobno spoznamo s komponentami tehnološke platforme. Nato pripravimo načrt ciljne arhitekture in se lotimo integracije posameznih komponent v delujočo oblačno infrastrukturo. Oblačno infrastrukturo preizkusimo tako z uporabniškega kot administratorskega vidika ter analiziramo njeno delovanje. Na koncu izpostavimo ugotovitve in ključne lastnosti te rešitve.

Ključne besede: računalništvo v oblaku, FI-PPP, FIWARE, IaaS, OpenStack, zasebni oblak.

Abstract

More and more companies are choosing to migrate their information technology services to either public, private or hybrid cloud. Cost savings, flexibility and faster time to market are just some of the benefits, but we should not forget the dangers that such a model entails. One of the key barriers to migrating services to the cloud is vendor lock-in as different providers offer their own non-standardized data formats and applications. An important factor in the use of proprietary solutions are also expensive licensing fees. In this thesis we, therefore, examine FIWARE open technology platform, which represents an open alternative to well established proprietary solutions for building cloud infrastructures. The main goal is to build and evaluate an Infrastructure as a Service cloud on an open platform. Firstly, we define the concepts of cloud computing and examine the technologies behind it, where we focus on providing Infrastructure as a Service. We take a look at a FIWARE project and thoroughly examine components of the technology platform. After that we prepare a plan for the target architecture and integrate individual components into a working cloud infrastructure. We test and analyse our cloud infrastructure from the end-user as well as administrator perspectives. Finally, we highlight the findings and key features of this solution.

Keywords: cloud computing, FI-PPP, FIWARE, IaaS, OpenStack, private cloud.

Poglavje 1

Uvod

Eksponentno večanje informacijskih potreb in nenehne zahteve po večji učinkovitosti poslovnih procesov ter zmanjševanju stroškov na vseh področjih narekujejo nove trende informacijske tehnologije [8]. Vse več podjetij se tako odloča za selitev svojih informacijskih storitev v oblak. Računalništvo v oblaku je širok pojem, ki lahko označuje celoten nabor različnih storitev. Tehnološko gledano gre za način zagotavljanja in upravljanja storitev informacijskih tehnologij, pri čemer so različni viri združeni v homogeno celoto (konsolidacija virov) in ponujeni v obliki storitve, ki je na voljo na zahtevo s pomočjo samopostrežnih portalov [9]. Pomembno je, da lahko do storitev dostopamo na standardne načine preko omrežja kjerkoli in kadarkoli ter za storitev plačujemo po porabi. Odvisno od nivoja abstrakcije ponujenih virov govorimo o različnih storitvenih modelih. Ponujamo lahko storitve infrastrukture, torej shrambo, omrežja in strežnike, kar imenujemo infrastruktura kot storitev (*angl.* Infrastructure as a Service - IaaS), aplikacijsko platformo oziroma platformo kot storitev (*angl.* Platform as a Service - PaaS) ali pa celotno aplikacijo, ki teče pri ponudniku storitev, čemur pravimo programska oprema kot storitev (*angl.* Software as a Service - SaaS). Glede na tip namestitve ločimo še javne, zasebne, hibridne in skupnostne oblake.

Kadar želimo v podjetju ohraniti popoln nadzor nad podatki, se odločimo za vzpostavitev zasebnega oblaka, ki omogoča višji nivo varnosti in zasebnosti [10]. Zmanjševanje stroškov, prilagodljivost in hitrejša izvedba so zgolj nekatere od prednosti, ki jih lahko nudi računalništvo v oblaku, vendar ob tem ne smemo pozabiti na nevarnosti, ki jih tak model prinaša. Ena izmed ključnih ovir pri selitvi storitev v oblak je zaklepanje uporabnikov znotraj ponudnika, saj različni ponudniki ponujajo lastne nestandardizirane formate podatkov in aplikacij [11]. Pomemben faktor pri uporabi lastniških rešitev predstavlja tudi visok strošek licenčnin. V diplomskem delu se bomo zato lotili odprte tehnološke platforme FIWARE, ki se razvija v okviru javno-zasebnega partnerstva za internet prihodnosti (*angl.* Future Internet Public-Private Partnership Programme - FI-PPP) Evropske komisije in predstavlja odprto alternativo uveljavljenim lastniškim rešitvam za postavitev oblačne infrastrukture. Cilj diplomske naloge je vzpostavitev in evalvacija zasebnega oblaka na odprti platformi FIWARE z namenom nudenja infrastrukture kot storitve.

V prvem delu bomo natančneje definirali koncepte računalništva v oblaku ter tehnologije, ki to omogočajo, nato bomo predstavili projekt FIWARE in tehnološko platformo, ki je jedro projekta. Podrobneje se bomo lotili komponent, ki omogočajo nudenje infrastrukture kot storitve in preučili njihovo delovanje. Pripravili bomo načrt ciljne arhitekture in z integracijo posameznih komponent v našem okolju vzpostavili delujočo oblačno infrastrukturo. Preizkusili jo bomo tako z uporabniškega kot administratorskega vidika ter analizirali njeno delovanje. Opisali bomo postopek vzpostavitve ter prikazali uporabo, upravljanje in možnosti za selitev obstoječih aplikacij v naš oblak. Na koncu bomo izpostavili ugotovitve in ključne lastnosti te rešitve.

Poglavje 2

Računalništvo v oblaku

2.1 Definicija

Kot smo omenili že v uvodu, je računalništvo v oblaku zelo širok pojem. Če se navežemo na definicijo Nacionalnega inštituta za standarde in tehnologijo (*angl.* National Institute of Standards and Technology - NIST) [9], je računalništvo v oblaku model omrežnega dostopa do deljenih računalniških virov (omrežje, strežniki, shramba, aplikacije in storitve), ki so lahko hitro oskrbovane in izdane z minimalnim upravljavskim trudom oziroma z minimalno interakcijo ponudnika storitve. Tak model ima pet glavnih značilnosti, in sicer [9]:

1. **Samopostrežba na zahtevo** (*angl.* On-demand self-service)
   Uporabnik oblačnih storitev si lahko kadarkoli sam ustvari in prilagaja vire, ki jih potrebuje, običajno preko samopostrežnih portalov, predvsem pa brez potrebe po udeležbi osebja, ki oblak upravlja.

2. **Omrežni dostop** (*angl.* Broad network access)
   Vsi viri so na voljo preko standardnih omrežij in kadarkoli enostavno dosegljivi preko standardnih protokolov širokemu naboru uporabnikov in uporabniških naprav.

3. **Agregacija virov** (*angl.* Resource pooling)
   Računalniški viri (shramba, pomnilnik, procesor, pasovna širina) so združeni v homogeno celoto in ponujeni različnim uporabnikom kot bazen virov v večnajemniškem modelu (*angl.* Multi-tenancy). Uporabniki storitev oblaka večinoma ne moremo vedeti, na katerem fizičnem viru se izvaja naš proces.

4. **Elastičnost** (*angl.* Rapid elasticity)
   Uporabnik oblačnih storitev lahko hitro pridobi nove vire, če jih potrebuje, oziroma vrne neporabljene vire oblaku, lahko pa se skaliranje virov odvija tudi popolnoma avtomatsko. Uporabnik ima občutek neskončnosti virov glede na svoje potrebe.

5. **Merjena storitev** (*angl.* Measured service)
   Oblačni sistem samodejno beleži podatke o porabi virov, ki so transparentno dostopni tako uporabniku kot tudi upravljavcu sistema. Na ta način lahko uporabniku zaračunamo storitev po dejanski porabi, merjene količine pa so v veliki meri odvisne od ponujene storitve oziroma nivoja abstrakcije (število aktivnih uporabnikov, procesorska moč, količina shranjenih podatkov).

2.2 Namestitveni modeli

Namestitveni modeli določajo tip namestitve oblačne infrastrukture oziroma nadzornika virov v njej in ciljne uporabnike, katerim so storitve dostopne. Organizacija, ki ima nadzor nad viri, ima v vsakem trenutku vpogled v delovanje virov, tehnično ima dostop do vseh podatkov na virih in upravlja z oblačnimi viri, medtem ko je množica uporabnikov z dostopom do virov lahko različna. Poznamo štiri osnovne namestitvene tipe, in sicer javni oblak, skupnostni oblak, zasebni oblak in hibridni oblak [9].

2.2.1 Javni oblak

Javni oblak (*angl.* Public cloud) je namestitveni model, pri katerem je oblačna infrastruktura v lasti zunanjega ponudnika, njene storitve pa so dostopne splošni javnosti. Vsi viri so konsolidirani in deljeni med različnimi uporabniki v večnajemniškem modelu. Pri tem se uporabljajo različni varnostni mehanizmi za ločitev posameznih stanovalcev oziroma najemnikov (*angl.* Tenants), ki lahko dostopajo samo do svojih virov oziroma omogočijo omejen dostop do svojih virov tudi drugim uporabnikom.

Kombinacija večnajemniškega modela, ki povečuje izkoriščenost virov, in velika mera standardizacije ter avtomatizacije, ki sta prisotni v javnih oblakih, prinašata znatne prihranke. Ponujene storitve so lahko v določeni meri brezplačne, običajno pa se plačujejo po porabi, kar nam omogoča, da najamemo samo toliko virov, kot jih potrebujemo. Na ta način se znebimo stroška vzpostavitve lastne infrastrukture, operativni stroški pa so predvidljivi. Pomembno vlogo ima dogovor na ravni storitve (*angl.* Service-Level Agreement - SLA) med uporabnikom in ponudnikom, ki med drugim določa razpoložljivost in zmogljivost najetih virov.

Največja pomanjkljivost javnih oblakov je vprašanje varnosti in zaupnosti podatkov oziroma zaupanje med ponudnikom in uporabnikom storitev, saj upravljanje svojih podatkov zaupamo tretji osebi, stopnja varnosti pa se lahko precej razlikuje od ponudnika do ponudnika.

Javni oblak je najpogostejši tip oblaka, zato ga pogosto označujemo kar z besedo oblak. Primeri ponudnikov javnih oblakov so Amazon, Rackspace, Google, Microsoft, IBM, HP, VMware in Facebook.

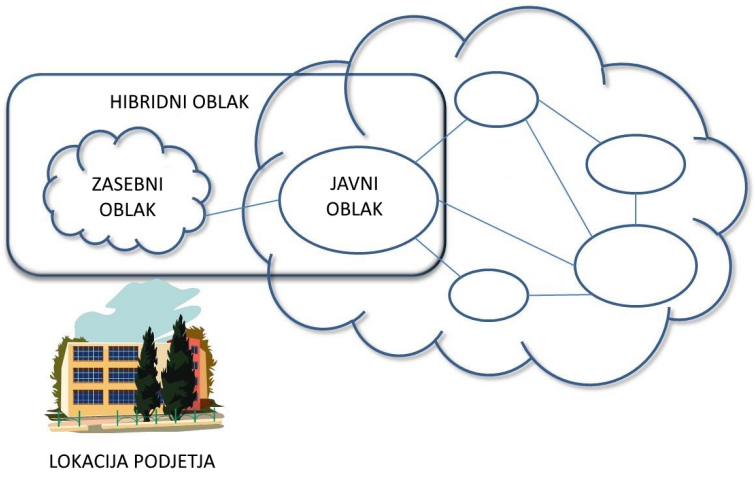
2.2.2 Skupnostni oblak

Skupnostni oblak (*angl.* Community cloud) predstavlja model, kjer je oblačna infrastruktura v lasti ene ali več organizacij, njene storitve pa so dostopne omejenemu krogu uporabnikov (skupnosti) s skupnimi cilji, varnostnimi zahtevami, politikami in standardi, ki jih morajo upoštevati. Običajno gre za

različne zdravstvene, znanstvene ali vladne organizacije, ki si na tak način delijo stroške vzpostavitve in upravljanja oblačne infrastrukture. Deljenje virov je podobno kot v javnem oblaku, vendar so lastnosti infrastrukture prilagojene ožjemu krogu uporabnikov. Infrastruktura skupnostnega oblaka se lahko nahaja v prostorih upravitelja ali na drugi lokaciji. Primer skupnostnega oblaka bo slovenski državni računalniški oblak, ki bo državnim institucijam omogočal računske, shranjevalne, razvojne, poslovne in druge storitve [12].

2.2.3 Zasebni oblak

Zasebni oblak (*angl.* Private cloud) je namestitveni tip pri katerem je oblačna infrastruktura v lasti organizacije in je namenjena za lastne potrebe. Še vedno podpira koncepte računalništva v oblaku, torej model v obliki storitev, samopostrežbo, oskrbovanje na zahtevo ter občutek neskončnosti virov. Običajno se infrastruktura nahaja v prostorih podjetja, vendar to ni nujno. Prav tako lahko upravljanje prepustimo zunanjemu ponudniku, ne glede na to pa z zasebnim oblakom lahko zagotovimo višji nivo varnosti in zasebnosti podatkov, saj je infrastruktura v celoti namenjena interni uporabi [10].



Slika 2.1: Shema hibridnega oblaka.

2.2.4 Hibridni oblak

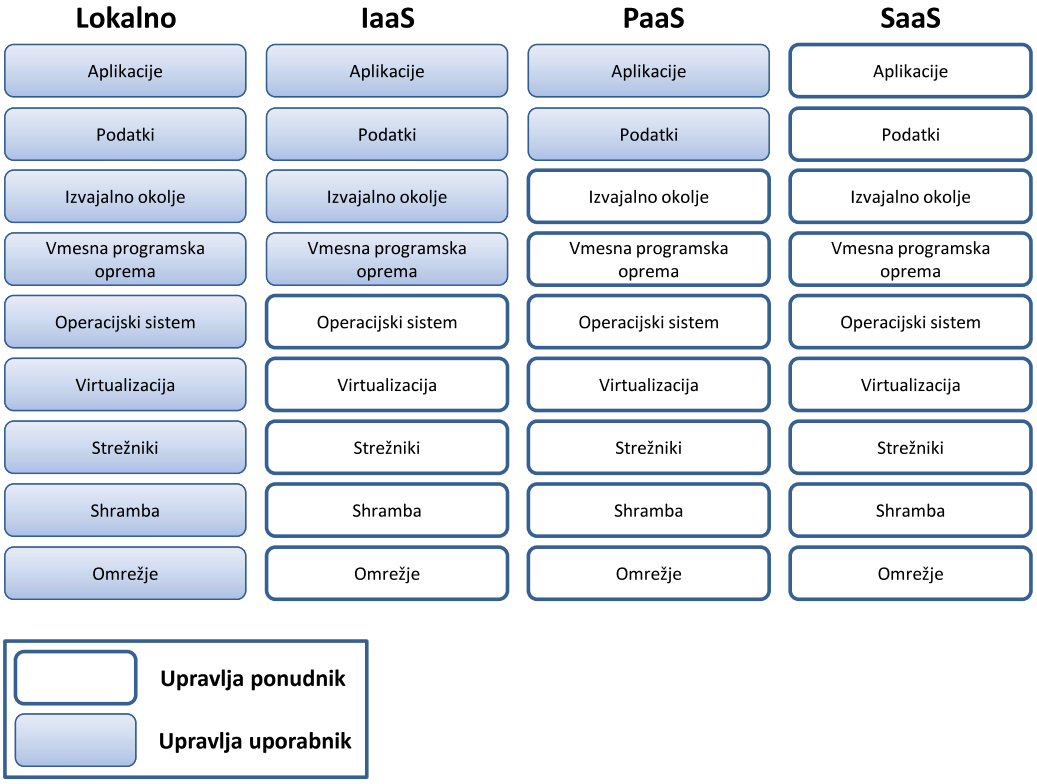
Hibridni oblak (*angl.* Hybrid cloud) predstavlja kombinacijo dveh ali več namestitvenih tipov (javnega, skupnostnega in zasebnega oblaka) [9] in tako transparentno omogoča prednosti več kot enega tipa. Največkrat gre za kombinacijo zasebnega in javnega oblaka kot prikazuje slika 2.1, pri čemer organizacije svoje procese razdelijo in del procesov zaradi varnostnih pomislekov, manjših zakasnitev ali posebnih zahtev gostijo v zasebnem oblaku, preostanek manj kritičnih procesov pa prenesejo v javni oblak, ki nudi cenovno učinkovitejšo in bolj elastično rešitev. Ker se zaupanje v javne oblake gradi postopoma, lahko hibridni oblak predstavlja vmesno točko pri selitvi procesov v javne oblake, pri čemer javni oblak za začetek izkoristimo zgolj za dinamično razvojno okolje ali shranjevanje varnostnih kopij [13]. Če organizacija vire javnega oblaka uporablja enakovredno kot vire zasebnega, vendar le takrat, ko jih v zasebnem oblaku primanjkuje, govorimo o razširjanju v oblak (*angl.* Cloudbursting) [14]. Največji izziv hibridnih oblakov je interoperabilnost in prenosljivost delovnih bremen med različnimi oblaki, ki zahteva kompleksne rešitve ali enotno platformo na obeh straneh [15].

2.3 Storitveni modeli

Kot smo že omenili, je ena od ključnih lastnosti računalništva v oblaku storitveni poslovni model oziroma ponudba virov v obliki storitev, od nivoja abstrakcije virov pa je odvisno, o katerem storitvenem modelu govorimo. Poznamo tri osnovne storitvene modele [9], in sicer: infrastrukturo kot storitev, platformo kot storitev in programsko opremo kot storitev. Posamezne nivoje abstrakcije lahko razberemo iz sheme na sliki 2.2.

V zadnjem času lahko srečamo še precej drugih XaaS terminov, ki bolj podrobno opredeljujejo ali pa predstavljajo nadgradnjo osnovnih storitvenih modelov, pri čemer X označuje določeno storitev npr. komunikacija kot storitev (*angl.* Communications as a Service - CaaS), namizje kot storitev (*angl.* Desktop as a Service - DaaS), nadomestni podatkovni center kot sto-

ritev (*angl.* Disaster Recovery Center as a Service - DRCaaS) ali celo IT kot storitev (*angl.* IT as a Service - ITaaS).



Slika 2.2: Storitveni modeli računalništva v oblaku [1].

2.3.1 Infrastruktura kot storitev

Infrastruktura kot storitev je najbolj osnoven storitveni model računalništva v oblaku, ki vključuje zmožnost oskrbovanja računskih, podatkovnih in omrežnih kapacitet ter drugih računalniških virov. To pomeni, da si lahko uporabnik na zahtevo ustvari navidezne stroje oziroma instance z določeno procesorsko močjo, prostorom in omrežno povezavo. Nanje lahko namesti poljubno programsko opremo, ki vključuje operacijski sistem in aplikacije. Infrastrukture oblaka ne upravlja in nadzira, saj je ta v domeni ponudnika storitve,

ki mora poskrbeti za objekt (običajno je teh več) z ustreznim napajanjem, hlajenjem in fizičnim varovanjem, strežnike, shranjevalne sisteme, omrežno opremo, povezljivost v svet ter programsko opremo, ki omogoča poganjanje, upravljanje ter samopostrežbo navideznih strojev. Uporabnik ima tako nadzor hkrati pa tudi skrb za vzdrževanje operacijskega sistema, podatkov, aplikacij in dela omrežnih storitev [9, 15]. Uporabo se običajno zaračunava na minutni, urni ali mesečni ravni glede na kapaciteto virov, ki jih je uporabnik v tem času uporabljal. Tak model omogoča največjo fleksibilnost glede delovnih bremen, ki jih lahko poganjamo, v primerjavi s storitvenimi modeli na višjih nivojih.

Med največjimi ponudniki, ki omogočajo infrastrukturo kot storitev so Amazon s svojo Elastic Compute Cloud ponudbo [16], Google s Compute Engine storitvijo [17], Rackspace [18], DigitalOcean [19], Microsoft [20] in IBM [21]. Vedno več ponudnikov lahko najdemo tudi v Sloveniji med drugim NIL s storitvijo FlexIT [22], Optimus IT s storitvijo mojOblak [23] in storitev PosiTa Pošte Slovenije [24]. Rešitve, ki omogočajo postavitev zasebnega ali skupnostnega oblaka z namenom nudenja infrastrukture kot storitve vključujejo VMware vCloud Suite, odprtokodno platformo OpenStack, na kateri je osnovana tudi tehnološka platforma FIWARE, ogrodje OpenNebula, Eucalyptus in Microsoft System Center v kombinaciji z Microsoft Azure Pack.

V nadaljevanju se bomo osredotočili na ta storitveni model in konkretno rešitev za gradnjo oblačne infrastrukture z odprtokodno platformo FIWARE.

2.3.2 Platforma kot storitev

Naslednji storitveni model predstavlja platforma kot storitev, ki poleg storitev infrastrukture vključuje še izvajalno okolje ter različne aplikacijske storitve, kot so podatkovne baze. Omogoča postavitev različnih tipov aplikacij, ki so razvite z uporabo programskih jezikov in orodij ponujenih s strani ponudnika. Uporabnik ima pri tem nadzor zgolj nad podatki, postavljeno aplikacijo in določenimi nastavitvami gostujočega okolja, ne pa tudi nad spodaj

ležečo infrastrukturo [9]. Tak model razvijalcem omogoča, da se osredotočijo na razvoj aplikacije in se jim ni potrebno ukvarjati z infrastrukturo. Največji problem je zaklepanje uporabnikov znotraj ponudnika, saj različni ponudniki ponujajo različna izvajalna okolja in je potrebno aplikacije velikokrat prilagoditi, če jih želimo prenesti drugam.

Primeri različnih platform ponujenih kot storitev so Microsoft Azure [20], Google App Engine [25], RedHat Openshift [26] in Heroku [27]. Kot zanimivost lahko omenimo, da Heroku nima lastne infrastrukture za gostovanje aplikacij na njihovi platformi, temveč uporablja Amazonov IaaS Elastic Compute Cloud. Poleg javnih ponudnikov obstajajo različne rešitve za nudenje platforme kot storitve v lastnem oblaku. OpenShift lahko namestimo tudi v zasebnem oblaku, platforma FIWARE nudenje platforme kot storitve omogoča s pomočjo komponente Pegasus PaaS Manager [28], obstaja pa še nekaj drugih rešitev, kot je npr. CumuLogic, ki omogoča integracijo z različnimi ponudniki infrastrukture kot storitve.

2.3.3 Programska oprema kot storitev

Najvišji storitveni nivo predstavlja programska oprema kot storitev, ki omogoča izvajanje aplikacije v infrastrukturi oblaka. Aplikacija je dostopna različnim odjemalcem preko spletnih brskalnikov ali aplikacijskih vmesnikov. Celotno vzdrževanje prevzame ponudnik storitve. Uporabniku pri tem ni treba skrbeti za upravljanje ali nadzor spodaj ležeče infrastrukture oblaka, omrežja, strežnikov, operacijskega sistema, licenc ter individualnih aplikacijskih zmožnosti [9]. Na voljo ima delujočo aplikacijo v oblaku za katero običajno plačuje mesečno naročnino ali porabo določenih količin znotraj aplikacije, kot npr. količino shranjenih podatkov.

Ponudba programske opreme kot storitve je zelo široka, saj zajema različne aplikacije, od socialnih omrežij kot so Twitter, LinkedIn in Facebook, do programske opreme za upravljanje odnosov s strankami, kot je Salesforce CRM, ali spletne pošte, ki jo ponujajo Google, Yahoo, Hotmail in drugi.

2.4 Oblačna infrastruktura

Oblačna infrastruktura je zbirka strojne in programske opreme, ki zagotavlja značilnosti oblačnega modela, torej samopostrežbo na zahtevo, omrežni dostop, agregacijo virov, elastičnost in merjeno storitev [9].

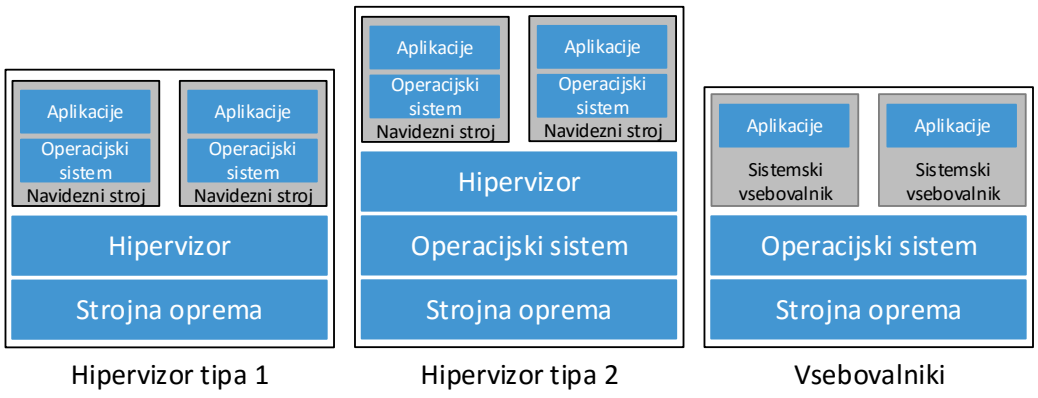
Strojno opremo sestavlja kopica fizičnih strežnikov, shranjevalnih sistemov in omrežnih komponent, ki so nameščeni v ustrezno opremljenih podatkovnih centrih. Večina ključnih komponent je običajno podvojenih za zagotavljanje nemotenega delovanja v primeru odpovedi katere izmed njih. Za pravilno delovanje in visoko razpoložljivost strojne opreme moramo v podatkovnih centrih zagotavljati ustrezno hlajenje, napajanje, fizično varovanje, protipožarno zaščito in nadzor, vedno več pozornosti pa se posveča tudi varčnosti in energetski učinkovitosti, ki vplivata na obratovalne stroške ter posledično ceno končne storitve. Ne smemo pozabiti niti na povezljivost v svet z zadostno pasovno širino in stopnjo redundance, ki nam omogočata omrežni dostop do virov.

V tem delu se osredotočamo predvsem na drugo komponento oblačne infrastrukture, torej programsko opremo, ki zagotavlja agregacijo in abstrakcijo teh virov ter ostale značilnosti oblaka na različnih storitvenih nivojih. Osnovna tehnologija, ki omogoča abstrakcijo virov na nivoju infrastrukture, je virtualizacija. Poleg virtualizacije strežnikov, shrambe in omrežij potrebujemo vsaj še skupek orodij, ki omogočajo avtomatizacijo dodeljevanja, razporejanja, ustvarjanja in brisanja posameznih virov, nadzor nad infrastrukturo, avtentikacijo in avtorizacijo uporabnikov ter nenazadnje vmesnik za samopostrežbo virov [15]. Konkretno rešitev na primeru platforme FIWARE bomo spoznali v drugem delu.

2.4.1 Virtualizacija strežnikov

Virtualizacija strežnikov je danes že povsem običajna praksa tudi v klasičnih strežniških okoljih, saj omogoča boljšo izkoriščenost virov. Gre za zmožnost deljenja virov strojne opreme (procesorja, pomnilnika, diska in omrežne po-

vezave) ter poganjanja več operacijskih sistemov na posameznem fizičnem sistemu. Operacijski sistem tako ne teče več neposredno na realni strojni opremi, ki ima vlogo gostitelja, temveč kot gost v izoliranem okolju imenovanem navidezni stroj (*angl.* Virtual Machine - VM) ali instanca, ki jo lahko v določenih primerih selimo med različnimi fizičnimi stroji tudi med samim delovanjem. Programska oprema, ki nam omogoča ta nivo abstrakcije, se imenuje hipervizor ali upravljalec navideznih strojev [29] in je zgolj ena od komponent oblačne infrastrukture. Kadar hipervizor teče neposredno na realni strojni opremi, govorimo o hipervizorjih tipa 1, kot so VMware ESXi, Microsoft Hyper-V, Oracle VM Server for x86, Xen in KVM (*angl.* Kernel-based Virtual Machine). V primeru ko hipervizor teče kot aplikacija na obstoječem operacijskem sistemu, npr. VMware Workstation ali Oracle VirtualBox na operacijskem sistemu Windows, je to hipervizor tipa 2. KVM je sicer del jedra splošno-namenskega operacijskega sistema Linux, vendar ga zaradi načina delovanja običajno uvrščamo med hipervizorje tipa 1 [30].



Slika 2.3: Primerjava hipervizorjev različnih tipov in vsebovalnikov.

Poleg običajne virtualizacije strojne opreme se počasi uveljavljajo tudi sistemski vsebovalniki (*angl.* Containers), ki omogočajo izolacijo sistema na nivoju operacijskega sistema. Iz uporabniškega vidika so vsebovalniki videti podobno kot navidezni stroji, saj imajo ločen procesni, omrežni in shrambni prostor, vendar si delijo skupno jedro operacijskega sistema. Gre za neke

vrste prenosljive peskovnike, v katerih lahko poganjamo aplikacije. Na ta način se znebimo simulacije strojne opreme in poganjanja več instanc operacijskega sistema, kar omogoča še bolj učinkovito izrabo virov, se pa pri tem omejimo na okolja, ki uporabljajo isto jedro kot gostujoči sistem. Operacijski sistem Linux podpira vsebovalnike že dalj časa [31], podpora za vsebovalnike pa prihaja tudi v operacijski sistem Windows Server s prihajajočo različico 2016 [32]. Primerjavo vsebovalnikov in hipervizorjev različnih tipov lahko vidimo na sliki 2.3.

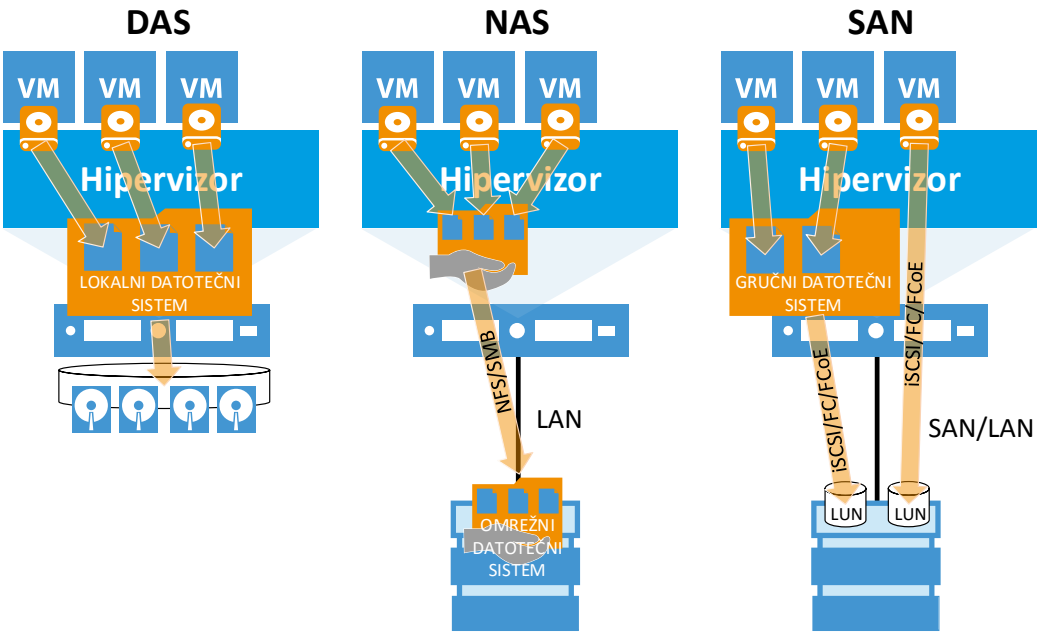
2.4.2 Virtualizacija shrambe

V poslovnih okoljih se večinoma uporabljajo namenski shranjevalni sistemi, ki omogočajo visoko razpoložljiv dostop do deljene shrambe več fizičnim strežnikom. S pomočjo tehnologij, kot je čezmerno polje samostojnih diskov (*angl.* Redundant array of independent disks - RAID), množico običajno trdih diskov ali pogonov brez premičnih delov (*angl.* Solid State Drive - SSD) združijo v eno ali več logičnih enot, iz katere se dodeljujejo shranjevalni viri in pri tem skrijejo fizične podrobnosti shranjevalnih enot.

Sistemi za omrežno dostopno shrambo podatkov (*angl.* Network-attached storage - NAS) tako s pomočjo protokolov, kot sta NFS (*angl.* Network File System) in SMB (*angl.* Server Message Block), omogočajo deljenje shrambe na nivoju datotečnega sistema preko omrežja IP (*angl.* Internet Protcol). Virtualizacijska platforma na fizičnih strežnikih nato na deljen datotečni sistem shranjuje navidezne diske v obliki datotek, ki jih dinamično ustvarja, briše ali celo povečuje po potrebi. Navidezni diski so predstavljeni navideznim strojem kot bločne enote.

Naslednja možnost je uporaba omrežij podatkovnih shramb (*angl.* Storage area network - SAN), kjer deljenje poteka na bločnem nivoju preko protokolov iSCSI (*angl.* Internet Small Computer System Interface), FCoE (*angl.* Fibre Channel over Ethernet) ali FC (*angl.* Fibre Channel). Na takšnem sistemu lahko ustvarimo logične shranjevalne enote (*angl.* Logical Unit - LUN), ki jih nato predstavimo fizičnim strežnikom preko omrežja.

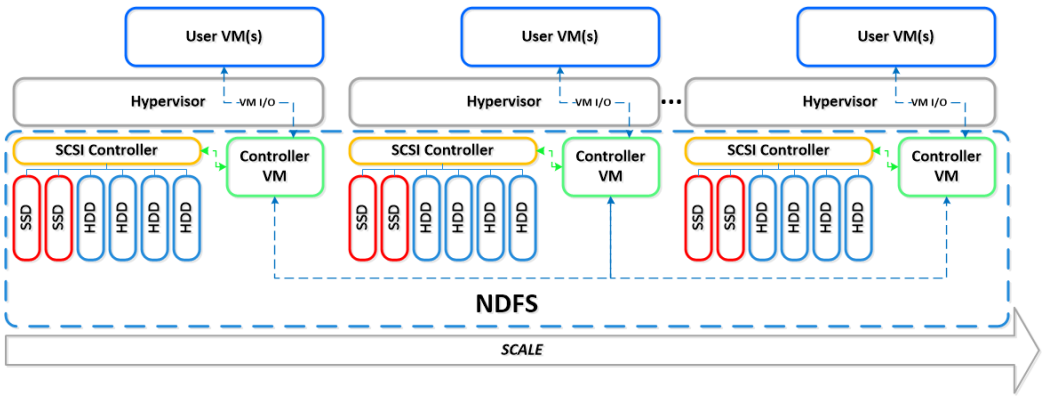
Hipervizor lahko na fizičnem strežniku posamezno logično enoto direktno predstavi navideznemu stroju kot bločno enoto (disk), vendar v ta namen potrebujemo za vsako bločno enoto navideznega stroja svojo logično enoto na shranjevalnem sistemu. Logična enota je lahko tudi skupna za več navideznih diskov. V tem primeru strežniki na posamezni logični enoti vzdržujejo gručni datotečni sistem, kot je VMFS (*angl.* VMware Virtual Machine File System), in ga uporabijo za shranjevanje navideznih diskov v obliki datotek, podobno kot pri sistemih NAS.



Slika 2.4: Primerjava shrambe DAS, NAS in SAN.

Namenski shranjevalni sistemi običajno predstavljajo precejšen strošek in imajo omejeno skalabilnost, zato veliki ponudniki oblačnih storitev, kot so Google, Amazon in Facebook uporabljajo drugačen pristop za shranjevanje podatkov [33]. Podatke shranjujejo s pomočjo porazdeljenih datotečnih sistemov, ki temeljijo na objektni shrambi in tečejo na gruči običajni strežnikov oziroma vozlišč povezanih preko omrežja. Podatki so ločeni od metapodatkov in replicirani med več vozlišči znotraj podatkovnega centra. Sistem je

namreč zasnovan tako, da predvideva odpoved katere koli od komponent in se temu tudi prilagaja. V določenih premerih se kopije podatkov replicirajo na sekundarno lokacijo za dodatno mero varnosti. Kapaciteto in zmogljivost takšnih sistemov je enostavno razširiti z dodajanjem novih vozlišč, zato so sposobni obdelati enormne količine podatkov. Princip delovanja je hitro posnemalo kar nekaj odprtokodnih rešitev, med katerimi so HDFS (*angl.* Hadoop Distributed File System), GlusterFS, Ceph in še mnoge druge, trend pa se počasi pojavlja tudi v poslovnih okoljih z rešitvami, kot so Nutanix [2], EMC ScaleIO [34] in VMware vSAN [35]. Na sliki 2.5 lahko vidimo, kako Nutanix uporabi lokalne shranjevalne enote v strežnikih in jih predstavi hipervizorjem kot enotno shrambo NAS ali SAN s pomočjo porazdeljenega datotečnega sistema imenovanega NDFS (*angl.* Nutanix Distributed File System).

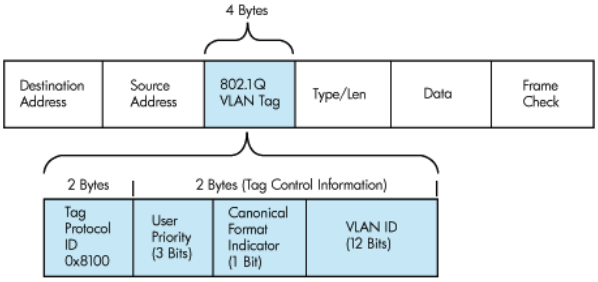


Slika 2.5: Nutanix NDFS [2].

V določenih primerih, kjer so podatki začasne narave, lahko za shranjevanje uporabimo tudi lokalne shranjevalne enote v samih strežnikih (*angl.* Direct Attached Storage - DAS). Na sliki 2.4 lahko vidimo primerjavo med shranjevanjem navideznih diskov na lokalnih diskih, sistemih NAS in sistemih SAN. Ne glede na to, kako shranjujemo podatke, oblačna infrastruktura običajno vključuje dodaten nivo abstrakcije v obliki upravljavca shrambe, ki uporabniku različne shrambe predstavi na enoten način [36].

2.4.3 Virtualizacija omrežij

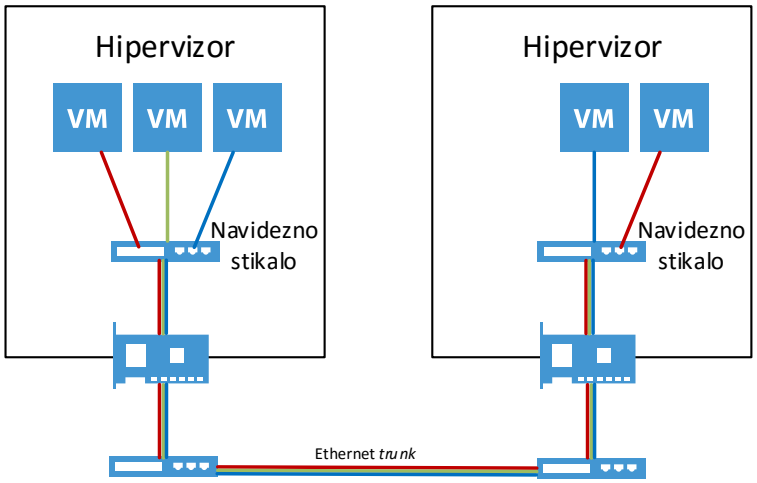
V nekaterih primerih zadostuje, da so navidezni stroji povezani v isto omrežje, običajno pa si želimo ločiti vsaj omrežja posameznih stanovalcev v večnajemniškem modelu ali pa jim celo omogočiti ustvarjanje izoliranih omrežij na zahtevo.



Slika 2.6: Zgradba okvirja Ethernet pri uporabi oznak 802.1Q [3].

Omrežja lahko ločimo z uporabo navideznih lokalnih omrežij (*angl.* Virtual Local Area Network - VLAN). Gre za ločitev omrežne infrastrukture na drugi plasti modela ISO/OSI (*angl.* International Organization for Standardization/Open Systems Interconnection model) na več podomrežij oziroma razpršitvenih domen (*angl.* Broadcast domain). Z vidika stikala lahko posamezno stikalo razdelimo na več izoliranih stikal tako, da vmesnike postavimo v različna navidezna lokalna omrežja, za povezavo med stikali pa vmesnike nastavimo v tako imenovan način trunk. To pomeni, da stikalo na tem vmesniku okvirje Ethernet iz različnih omrežij enolično označuje oziroma jih na podlagi oznake posreduje samo na ustrezne vmesnike. V ta namen se običajno uporablja protokol IEEE 802.1Q, kjer so okvirji navideznih lokalnih omrežij ločeni s posebnim 12-bitnim identifikatorjem omrežja, ki se vrine v zaglavje. To nam teoretično omogoča 4094 različnih navideznih omrežij, saj sta oznaki 0 in 4095 rezervirani, dejansko število pa je odvisno od podpore fizičnih stikal. Zgradbo okvirja pri uporabi 802.1Q lahko vidimo na sliki 2.6. V virtualiziranih okoljih ima hipervizor vlogo navideznega stikala, na katerega so transparentno povezani navidezni stroji, zato potrebujemo povezave trunk ne

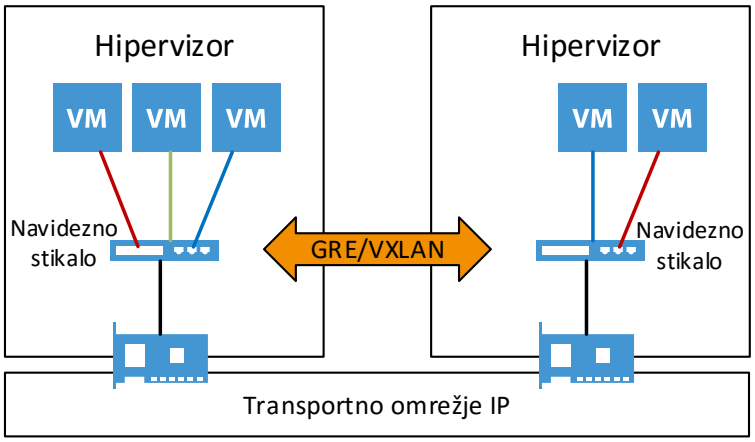
samo med fizičnimi stikali, temveč tudi do strežnikov. Dodajanje novega navideznega omrežja v tem primeru pomeni spremembe na fizični opremi, ki jih velikokrat ne moremo opraviti na zahtevo. Mobilnost navideznih strojev po drugi strani zahteva dostop do določenega omrežja iz katerega koli fizičnega strežnika, zato običajno v naprej pripravimo določen razpon navideznih lokalnih omrežij, ki jih pripeljemo do vseh strežnikov. Navidezni stroji lahko tako tečejo na poljubnem fizičnem strežniku in med seboj komunicirajo v izoliranem omrežju, kot vidimo na sliki 2.7.



Slika 2.7: Komunikacija preko navideznih lokalnih omrežij.

Poleg že omenjenih omejitev glede števila navideznih omrežij ima tak pristop še nekatere druge pomanjkljivosti. Celotno omrežje postane enotna domena na drugi plasti, ki je v osnovi namenjena komunikaciji sosednjih naprav in ima težave s skalabilnostjo. Protokol Ethernet namreč nima hierarhične zgradbe naslovov, ki bi omogočali agregacijo, ne vsebuje mehanizmov za preprečevanje zank in temelji na posredovanju okvirjev na podlagi učenja ter poplavljanju za neznane naslove. Nekatere od teh težav sicer lahko rešimo z uporabo kompleksnih protokolov kot sta 802.1aq in TRILL (*angl.* Transparent Interconnection of Lots of Links), saj omogočata uporabo podobnih usmerjevalnih algoritmov, kot se sicer uporabljajo na tretji plasti, vendar

imamo še vedno problem enotne razpršitvene domene, ki predstavlja enojno točko odpovedi. Napaka na enem delu omrežja lahko namreč povzroči izpad celotnega omrežja in s tem celotnega podatkovnega centra.



Slika 2.8: Komunikacija preko navideznih prekrivnih omrežij.

Boljši pristop predstavlja uporaba navideznih prekrivnih omrežij (*angl.* Overlay virtual networks), ki omogočajo podoben nivo abstrakcije omrežja, kot ga poznamo pri abstrakciji strojne opreme z virtualizacijo strežnikov. Na ta način lahko navidezna omrežja popolnoma ločimo od fizičnega omrežja. Vse, kar potrebujemo, je povezljivost strežnikov preko protokola IP, kar lahko precej poenostavi omrežno infrastrukturo in omogoča njeno skalabilnost. Navidezna stikala v strežnikih nato poskrbijo za izolacijo prometa med posameznimi navideznimi omrežji in njegovo enkapsulacijo za transport do ciljnega strežnika preko omrežja IP. Ločevanje prometa je odvisno od uporabljenega tunelskega protokola, ki je običajno VXLAN (*angl.* Virtual Extensible Local Area Network), GRE (*angl.* Generic Routing Encapsulation) ali STT (*angl.* Stateless Transport Tunneling), in omogoča povezljivost med navideznimi stroji na drugi ali tretji plasti ne glede na fizično topologijo omrežja, kot prikazuje slika 2.8. Protokoli za enkapsulacijo uporabljajo za ločevanje 24 in več bitne enolične identifikatorje. To omogoča bistveno večje število navideznih omrežij kot pri uporabi navideznih lokalnih omrežij. Ker so omrežja v celoti

definirana programsko, jih lahko enostavno dodajamo in brišemo na zahtevo, kar je zelo pomemben aspekt oblačne infrastrukture.

2.4.4 Virtualizacija omrežnih storitev

Poleg samih omrežij lahko virtualiziramo tudi različne omrežne storitve, kot so delilniki bremena, sistemi za zaznavo vdorov in požarne pregrade. Govorimo o virtualizaciji omrežnih funkcij (*angl.* Network functions virtualization - NFV), kjer funkcionalnost namenskih omrežnih naprav realiziramo s pomočjo programske opreme v obliki navideznih strojev, sistemskih vsebovalnikov ali druge tehnologije, ki lahko teče na običajnih strežnikih [37]. To nam precej poenostavi izstavitev različnih omrežnih storitev na zahtevo in njihovo upravljanje, saj ne potrebujemo namenske fizične opreme za različne storitve, temveč omrežne funkcije zaganjamo na virtualni infrastrukturi enako kot vse ostale aplikacije oziroma navidezne stroje. Omrežne storitve lahko enostavno ponudimo v upravljanje uporabnikom. Vsak najemnik lahko dobi svoj navidezni usmerjevalnik ali požarno pregrado, nad katero ima popoln nadzor. Iz vidika uporabnika je to dobrodošlo, ker lahko po potrebi celoten aplikacijski sklad skupaj z omrežnimi storitvami (požarnimi pregradami, delilniki bremena, ...), ki bi jih sicer moral nastavljati ponovno, preseli drugam v obliki navideznih strojev.

Zavedati se moramo, da imajo takšne rešitve tudi določene omejitve zlasti glede prepustnosti. Za posredovanje paketov z visoko hitrostjo so še vedno precej bolj učinkovite strojne rešitve. Tak primer so stikala v podatkovnih centrih, ki imajo prepustnost 1 Tb/s in več, vendar običajno služijo zgolj za transport in je njihova konfiguracija lahko dokaj statična. Na drugi strani je večina namenskih naprav, ki dela na višjih plasteh modela ISO/OSI (L4-L7), tako ali tako osnovana na običajnih strežnikih x86. Največji problem predstavlja implementacija sklada TCP/IP v operacijskem sistemu. Običajno je to različica Linuxa, kjer lahko dosežemo hitrosti do 1 Gb/s s posameznim procesnim jedrom. Če uporabimo lastno implementacijo sklada TCP/IP oziroma prosto dostopen Intelov razvojni paket za podatkovno ravnino (*angl.*

Data Plane Development Kit), ki ga danes uporablja večina komercialnih rešitev, lahko z ustrezno implementirano programsko opremo dosežemo hitrosti do 10 Gb/s na fizično jedro običajnega strežnika. Nekoliko bolj zahtevne so naprednejše požarne pregrade. Požarne pregrade nove generacije Palo Alto tako potrebujejo do 4 jedra za prepustnost 1 Gb/s [38], vendar gre za zelo zahtevno filtriranje prometa. Tudi takšne hitrosti običajno zadostujejo, saj imamo lahko namesto centralne požarne pregrade več manjših, ki ščitijo posamezno aplikacijo ali celo instanco. Večina večjih ponudnikov omrežne opreme tako danes ponuja tudi virtualne rešitve, obstaja pa tudi kar nekaj odprtokodnih rešitev, ki jih lahko uporabimo v oblačni infrastrukturi.

Poleg običajnih požarnih pregrad, ki so zgolj ponujene v virtualni namesto fizični obliki, poznamo še požarne pregrade na nivoju mrežnih kartic posameznih instanc. Gre za paketne filtre, ki so implementirani med navidezno mrežno kartico instance in navideznim stikalom v hipervizorju. Običajno preko vmesnika uporabnik določi kateri varnostni skupini pripada posamezna instanca in pravila komunikacije med skupinami, orkestracijski sistem pa nato poskrbi za pravila, ki morajo biti dodana v požarne pregrade pred vsako instanco. To omogoča precejšno fleksibilnost, skalabilnost in enostavno upravljanje, saj se lahko znebimo centralnih požarnih pregrad z množico pravil.

Poglavje 3

Projekt FIWARE

3.1 Kaj je FIWARE?

FIWARE je odprta pobuda Evropske komisije v sklopu javno-zasebnega partnerstva za internet prihodnosti, katerega cilj so večja učinkovitost poslovnih procesov in pametnejša infrastruktura, ki podpira inovativne aplikacije v različnih sektorjih [39]. Podjetjem in vladam poskuša pomagati z razvojem internetnih rešitev, ki bodo zmožne obdelovati eksponentno povečane spletne podatke, s katerimi se srečujemo v današnjem svetu in jim obstoječe tehnologije niso več kos. Projekt FIWARE tako ustvarja trajnostni, globalen in odprt ekosistem inovacij z namenom, da podjetnikom olajša uresničitev idej in priložnosti novega vala digitalizacije. Projekt temelji na petih stebrih, in sicer: tehnološki platformi FIWARE, prosto dostopnem testnem okolju FIWARE Lab, zbirki orodij za upravljanje FIWARE Ops, programu pospeševalcev FIWARE Accelerate in programu za promocijo pobude izven Evrope FIWARE Mundus [40].

3.1.1 Platforma FIWARE

Tehnološka platforma FIWARE, ki predstavlja jedro projekta, je razširitev odprte in hitro razvijajoče se oblačne platforme OpenStack. Poleg osnovnih možnosti gostovanja v oblaku ponuja dodano vrednost v obliki množice storitveno usmerjenih osnovnih gradnikov (*angl.* Generic Enabler - GE), ki zagotavljajo odprte aplikacijske programske vmesnike (*angl.* Application Programming Interface - API) za enostavnejši razvoj inovativnih aplikacij. Tako prinaša komponente, ki med drugim omogočajo povezavo z internetom stvari (*angl.* Internet of Things - IoT), podporo pametnim in kontekstno odvisnim aplikacijam s pomočjo procesiranja velike količine podatkov v realnem času, kot tudi analizo velikih podatkov (*angl.* Big Data) in vključitev naprednih spletnih vmesnikov. Specifikacije vseh komponent oziroma osnovnih gradnikov so javno dostopne, kar omogoča različne implementacije posameznih ponudnikov, hkrati so podprte z odprtokodnimi referenčnimi implementacijami, ki so na voljo v katalogu FIWARE in omogočajo takojšno uporabo s tem pa hitrejši prodor na trg. Če povzamemo, so glavne prednosti platforme FIWARE predvsem:

- hitrejši razvoj pametnih aplikacij s pomočjo odprtih aplikacijskih vmesnikov, ki so neodvisni od ponudnika gostovanja,
- odprtost platforme brez zaklepanja uporabnikov,
- enostavnejša integracija lastnih rešitev,
- fleksibilnost.

Z vidika ponudnika gostovanja v oblaku platforma FIWARE predstavlja alternativo uveljavljenim rešitvam kot:

- cenovno učinkovitejša rešitev od lastniških produktov,
- razširitev oblačne platforme OpenStack s funkcionalnostmi, ki naj bi omogočale poganjanje širšega nabora bremen:
  - visoka razpoložljivost virtualnih strojev oziroma instanc,

  - naprednejši razporejevalnik za boljšo izrabo virov,
  - podpora zagotavljanju kvalitete storitve bremen,
  - možnost določanje politike razporejanja instanc,
  - enotno upravljanje heterogenih okolij.

Pri tem se nam porajajo določeni dvomi oziroma morebitne slabosti, in sicer:

- kratek izdajni cikel in velike spremembe platforme OpenStack, ki prinašajo težavo s kompatibilnostjo razširitev v platformi FIWARE,
- odsotnost zagotovljene podpore pri postavitvi in upravljanju,
- dodatno delo potrebno za zagotovitev optimalnega delovanja,
- dovršenost komponent v trenutni fazi.

3.1.2 FIWARE Lab

FIWARE Lab je delujoči primerek platforme FIWARE, realiziran s pomočjo referenčnih implementacij komponent na množici geografsko porazdeljenih podatkovnih centrov oziroma vozlišč FIWARE (*angl.* FIWARE nodes) [41]. Zagotavlja prosto dostopno testno okolje, kjer lahko razvijalci preizkusijo svoje aplikacije na resničnih podatkih in uporabnikih, hkrati pa trgu predstavijo svoje ideje. Trenutno infrastruktura obsega 18 vozlišč [42] in kot taka lahko služi različnim potrebam širokega nabora uporabnikov. Dostop do virov je mogoč z enostavno registracijo na samopostrežnem portalu.

3.1.3 FIWARE Ops

Za lažjo vzpostavitev in upravljanje infrastrukture FIWARE je na voljo skupek orodij pod imenom FIWARE Ops [43]. Osnova je obstoječe orodje Mirantis Fuel, ki omogoča avtomatizirano postavitev okolja OpenStack, v katero poskušajo vključiti preostale komponente platforme FIWARE. Na ta način želijo poenostaviti vzpostavitev novih vozlišč in federacijo teh v skupno testno okolje FIWARE Lab.

3.1.4 FIWARE Accelerate

Najobetavnejše skupine in poslovne predloge finančno podpira in mentorira mreža evropskih organizacij s programom pospeševalcev FIWARE (*angl.* FIWARE Accelerate), v katerega je vključenih 16 pospeševalcev na različnih področjih. Skupno bodo majhnim in srednjim podjetjem, startupom ter spletnim podjetjem, ki uporabljajo tehnologijo FIWARE za razvoj inovativnih aplikacij v okviru projekta, razdelili 80 milijonov EUR evropskih sredstev [44].

3.1.5 FIWARE Mundus

Kljub temu, da gre za evropski projekt, ima FIWARE globalne ambicije, zato za promocijo pobude izven Evrope skrbi program FIWARE Mundus [45]. S širitvijo na globalnem nivoju vključuje podobne raziskave in inovacije drugod po svetu, ki so pomembne za dolgoročni obstoj projekta. Glavni cilj je vzpostavitev ne samo evropskega, temveč tudi globalnega ekosistema za internet prihodnosti, ki naj bi vodil k izboljšanju kakovosti življenja posameznikov in celotne družbe.

3.2 Koncept modularnih osnovnih gradnikov

3.2.1 Definicija

Posamezne komponente, ki sestavljajo referenčno arhitekturo tehnološke platforme FIWARE se imenujejo osnovni gradniki (*angl.* Generic Enablers) [46]. Gre za večkratno uporabne module s funkcionalnostmi, ki so skupne različnim področjem uporabe, v nasprotju z namenskimi gradniki (*angl.* Specific Enabler), ki so specifični zgolj za ozko področje uporabe [47]. Osnovni gradniki so storitveno usmerjeni in zagotavljajo odprte aplikacijske programske vmesnike. Aplikacijski programski vmesniki so namenjeni enostavnejšemu razvoju inovativnih aplikacij, kakor tudi interakciji posameznih osnovnih gradnikov s preostalimi osnovnimi gradniki [48]. Osnovni gradniki so razdeljeni v različna tehnična področja [49], in sicer:

- **Gostovanje v oblaku** (*angl.* Cloud Hosting) - združuje temeljne komponente za zagotavljanje in upravljanje računskih, shranjevalnih in omrežnih virov [50], kamor med drugim spadata samopostrežni portal (*angl.* Cloud Portal GE) in komponenta za upravljanje z virtualnimi stroji (*angl.* IaaS Resource Management GE), ki ju bomo podrobneje obravnavali v nadaljevanju dela.
- **Upravljanje s podatki** (*angl.* Data/Context Managment) - zajema gradnike, ki omogočajo kontekstno odvisnim aplikacijam učinkovito zbiranje, dostop, izmenjavo, procesiranje in analizo velike količine podatkov [51]. Eden izmed gradnikov je komponenta Cosmos za analizo velikih podatkov (*angl.* BigData).
- **Aplikacije** (*angl.* Applications) - predstavlja ekosistem za razvoj, objavo, upravljanje in uporabo aplikacij skozi celoten življenjski cikel, ki zajema tako tehnične kot tudi poslovne vidike [52]. Primer gradnika je tržnica za aplikacije in storitve ponujene v okviru FIWARE, ki se imenuje WMarket.

- **Internet stvari** (*angl.* Internet of Things - IoT) - sestavlja zaledni sistem ter vmesnike za interakcijo z resničnimi predmeti na enoten način [53].
- **Vmesniki za omrežja in naprave** (*angl.* Interface to Networks and Devices - I2ND) - področje zajemajo gradniki za odprto in standardizirano omrežno infrastrukturo [54].
- **Varnost** (*angl.* Security) - vsebuje mehanizme za zagotavljanje varnosti in zasebnosti ponujenih storitev [55]. V to področje spada komponenta za upravljanje z identitetami KeyRock.
- **Napredni spletni uporabniški vmesniki** (*angl.* Advanced Web UI) - področje združuje komponente za gradnjo inovativnih uporabniških vmesnikov z novimi vnosnimi metodami in interaktivnimi zmožnostmi, kot je interaktivna 3D grafika [56].

Cilj projekta FIWARE je predvsem identifikacija in specifikacija osnovnih gradnikov skupaj z razvojem in demonstracijo referenčnih implementacij. Odprte specifikacije osnovnih gradnikov (*angl.* GE Open Specifications) so prosto dostopne in vključujejo vse potrebne informacije za implementacijo osnovnih gradnikov, kar tipično zajema vsaj [57]:

- opis funkcionalnosti, področja in namena uporabe,
- razlago uporabljenih pojmov in kratic,
- definicije aplikacijskih programskih vmesnikov, ki jih mora gradnik ponujati za interakcijo z ostalimi komponentami,
- opis protokolov za interakcijo z ostalimi gradniki, aplikacijami ali zunanjimi storitvami.

Vsaka rešitev, ki implementira funkcionalnosti opisane v specifikaciji, je implementacija osnovnega gradnika, medtem ko se implementacija, ki je razvita v okviru projekta FIWARE in objavljena v katalogu, imenuje referenčna implementacija. Posamezne instance FIWARE (*angl.* FIWARE Instance) tako

integrirajo več osnovnih gradnikov oziroma njihovih implementacij, ki so bodisi referenčne ali lastne, in tako zagotavljajo različne storitve, odvisno od množice uporabljenih gradnikov. Tovrstna arhitektura nam omogoča veliko mero prilagodljivosti tehnološke platforme glede na potrebe.

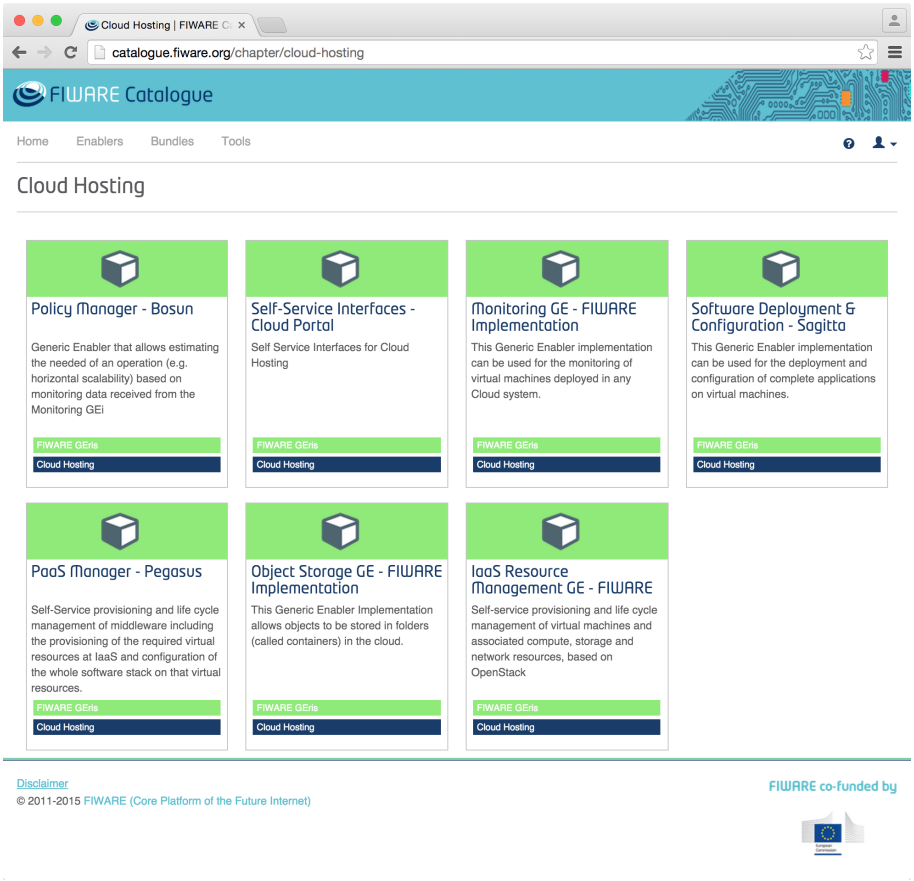
3.2.2 Razvoj

Začetna množica osnovnih gradnikov oziroma njihove specifikacije so bile določene v prvi fazi projekta glede na cilje projekta in primere uporabe različnih partnerjev [49]. V tej fazi se je začel tudi razvoj identificiranih gradnikov, hkrati pa so preko namenskih projektov sprejemali povratne informacije in zahteve po dodatnih funkcionalnostih, na podlagi katerih so identificirali nove osnovne gradnike [58]. Razvoj teh osnovnih gradnikov je potekal s pomočjo dveh odprtih klicev, v katerih so lahko sodelovali novi partnerji. Trenutno je razvoj osnovnih gradnikov omejen zgolj na člane projekta, vendar je v načrtu odprtje projekta za širšo skupnost, ki je pomembna za nadaljnji razvoj in obstoj. V ta namen bodo oblikovali pravila za upravljanje in postopek odobritve novih rešitev, ki naj bi vseboval različne zahteve glede arhitekture, dokumentacije, objave kode ter licenčnih pogojev.

Večina osnovnih gradnikov temelji na obstoječih odprtokodnih ali lastniških rešitvah partnerjev, ki sodelujejo v projektu. Osnova za gradnike na področju gostovanja v oblaku so tako komponente, ki jih ponuja odprtokodna oblačna platforma OpenStack in obstoječi vtičniki za omenjeno platformo [59]. Glavni namen je integracija posameznih rešitev v koherentno platformo za internet prihodnosti ter razširitve funkcionalnosti na podlagi analize potreb različnih uporabnikov [60]. Razvoj poteka s pomočjo agilnih metod, pri čemer seznam zahtev izhaja iz odprtih specifikacij posameznih osnovnih gradnikov, vendar zgolj za nove funkcionalnosti [61]. Posamezne iteracije trajajo mesec dni, vsaki tretji iteraciji pa sledi manjša izdaja s posodobitvijo kataloga in testnega okolja FIWARE Lab. Razvijalci za urejanje seznama zahtev, planiranje iteracij, sledenje napredku in medsebojno komunikacijo uporabljajo fleksibilno orodje JIRA Agile.

3.2.3 Objava v katalogu FIWARE

Katalog FIWARE je centralni repozitorij, kjer so objavljene referenčne implementacije osnovnih gradnikov s povezavami na odprte specifikacije, izvorno kodo, dokumentacijo, uporabniška navodila, licenčne pogoje in ostale relevantne informacije [62]. Primer objavljenih gradnikov s področja gostovanja v oblaku lahko vidimo v katalogu FIWARE na sliki 3.1. Objave so zaenkrat



Slika 3.1: Katalog FIWARE.

omejene zgolj na člane projekta FIWARE, vendar naj bi se to v prihodnje spremenilo [63]. Skrbniki posameznih gradnikov po potrebi oziroma ob vsaki manjši izdaji posodabljajo informacije v katalogu preko skrbniškega portala.

3.3 Upravljanje z identitetami - KeyRock

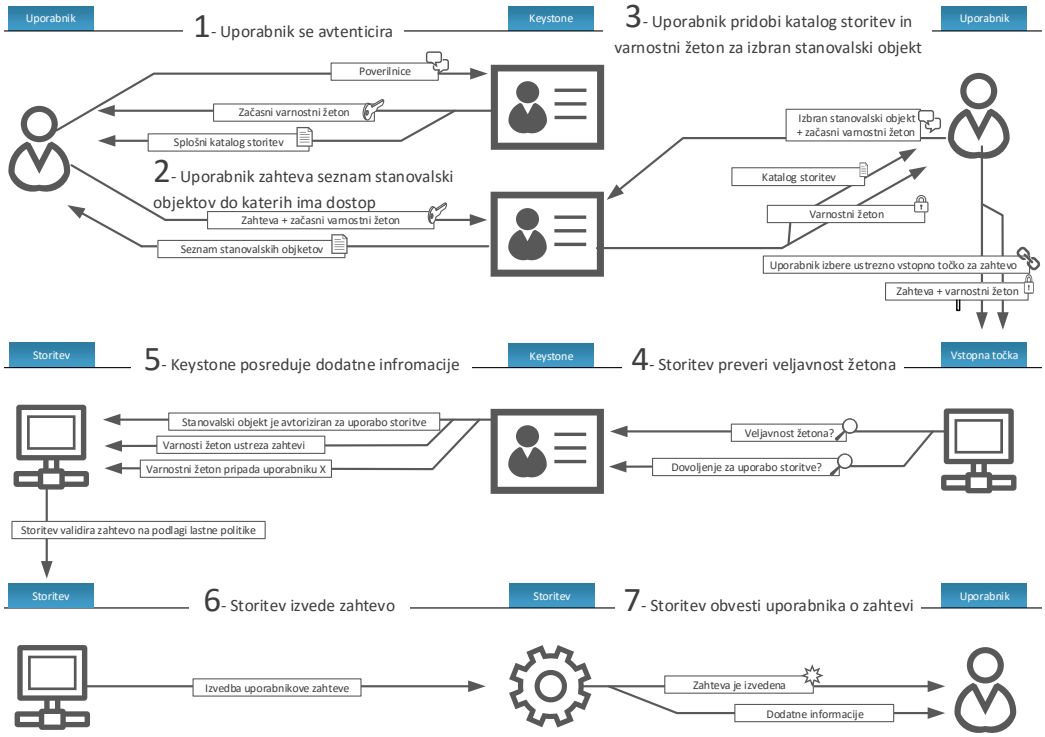
KeyRock oziroma komponenta za upravljanje z identitetami je osnovna komponenta platforme, ki omogoča upravljanje z uporabniki, njihovimi pravicami, stanovalskimi objekti ter storitvami, ki so na voljo [63]. Referenčna implementacija je sestavljena iz dveh delov: iz zalednega sistema Keystone in grafičnega uporabniškega vmesnika za administracijo imenovanega Horizon.

3.3.1 Keystone

Zaledni del upravljanja z identitetami je razširitev komponente OpenStack Keystone. Omogoča enoten sistem prijave z avtentikacijo uporabnikov in kontrolo dostopa. Podatke o uporabnikih varno hrani v relacijski podatkovni bazi, običajno je to MySQL, ali pa jih pridobiva iz zunanjih virov s protokoloma LDAP (*angl.* Lightweight Directory Access Protocol) ali OAuth. Uporabniki se lahko avtenticirajo s pomočjo poverilnic preko standardnega programskega vmesnika z uporabo protokola HTTP (*angl.* HyperText Transfer Protocol), pri čemer dobijo časovno omejen varnostni žeton (*angl.* Token) za komunikacijo z vsemi ostalimi komponentami. Te nato za vsako zahtevo s pomočjo komponente Keystone preverijo veljavnost žetona in po potrebi pridobijo ostale uporabniške atribute. Poleg podatkov o uporabnikih Keystone hrani tudi katalog vseh storitev in vstopnih točk njihovih aplikacijskih programskih vmesnikov, kar ostalim komponentam omogoča enostavno odkrivanje storitev, ki so na voljo [4].

Primer interakcije s komponento Keystone za izvedbo generične zahteve lahko vidimo na sliki 3.2. Uporabnik oziroma odjemalec se avtenticira tako, da pošlje zahtevo za varnostni žeton in pri tem posreduje ustrezne poverilnice. Od komponente Keystone prejme splošni katalog storitev in začasni varnostni žeton, ki ga uporabi za poizvedbo o stanovalskih objektih, do katerih ima dostop. Za izbran stanovalski objekt nato zahteva nov varnostni žeton, pri čemer prejme še katalog storitev, ki so mu na voljo. Iz kataloga iz-

bere ustrezno storitev in njeno vstopno točko, kateri posreduje svoje zahtevo in pri tem uporabi pridobljen varnostni žeton. Storitev nato pred izvedbo zahteve s pomočjo komponente Keystone preveri veljavnost žetona, dovoljenje za uporabo storitve in pridobi vse ostale potrebne atribute o uporabniku in stanovalskem objektu [64].



Slika 3.2: Procesni tok Keystone za izvedbo generične zahteve [4].

3.3.2 Horizon

Grafični uporabniški vmesnik za administracijo temelji na komponenti OpenStack Horizon. Gre za modularni spletni vmesnik implementiran na platformi Django, ki običajno teče na spletnem strežniku Apache s pomočjo modula mod_wsgi. Uporabnikom in administratorjem omogoča enostavno upravljanje z uporabniki in storitvami.

3.4 Upravljanje z virtualnimi stroji - DCRM

Komponenta za upravljanje z virtualnimi stroji oziroma DCRM omogoča gostovanje virtualnih strojev in upravljanje s pripadajočimi računskimi, shranjevalnimi in omrežnimi viri v podatkovnem centru. Rešitev temelji na odprti in hitro razvijajoči se oblačni platformi OpenStack [63]. Specifikacija komponente predvideva razširitev osnovnih možnosti gostovanja v oblaku s funkcionalnostmi, ki naj bi omogočale poganjanje širšega nabora bremen in boljšo izkoriščenost virov. V ta namen naj bi v komponento vključili napredni razporejevalnik, ki med drugim omogoča [65]:

- razporejanje virov na podlagi razširljivega nabora metrik vključno z dejansko zasedenostjo virov npr. procesorskega časa,
- dinamično prerazporejanje virov s pomočjo migracije virtualnih strojev med samim delovanjem,
- visoko razpoložljivost virtualnih strojev oziroma instanc,
- avtomatizirano izpraznitev posameznih računskih vozlišč z migracijo instanc na druga vozlišča za namen vzdrževanja,
- možnost določanja naprednih politik razporejanja instanc,
- podporo zagotavljanju kakovosti storitve (QoS) bremen.

Nekatere izboljšave, ki so bile razvite v okviru referenčne implementacije, so razvijalci tekom projekta prispevali skupnosti OpenStack. Ostale razširitve so zaenkrat opuščene oziroma so na voljo v okviru plačljive rešitve IBM Platform Resource Scheduler, ki je del produkta IBM Cloud Manager [66]. Prosto dostopna referenčna implementacija DCRM je tako skupek štirih projektov platforme OpenStack. To so OpenStack Glance, Nova, Cinder in Neutron [63]. Tehnološka platforma FIWARE zaenkrat temelji na predzadnji izdaji, to je OpenStack Juno.

3.4.1 Glance

Glance omogoča shranjevanje slik navideznih diskov in metapodatkov, ki se uporabljajo kot predloge za ustvarjanje novih instanc oziroma virtualnih strojev. Komponenta je sestavljena iz dveh procesov. Proces glance-api sprejema zahteve za odkrivanje, branje in shranjevanje slik preko programskega vmesnika REST, poskrbi za kopiranje slik v ustrezno shrambo in komunicira s procesom glance-registry, ki skrbi za procesiranje in shranjevanje metapodatkov v relacijski podatkovni bazi [4].

Komponenta za shranjevanje slik podpira različne tipe shramb, in sicer: lokalni datotečni sistem, bločno shrambo OpenStack Cinder, objektno shrambo OpenStack Swift, GridFS, Ceph, Amazon S3, Sheepdog in podatkovne shrambe na strežniku VMware ESXi ali vCenter [67]. Slike so lahko shranjene v različnih formatih, med drugim v surovem formatu in formatih QCOW2 (*angl.* QEMU Copy On Write), VMDK (*angl.* Virtual Machine Disk), VHD (*angl.* Virtual Hard Disk) ali ISO (*angl.* International Organization for Standardization 9660 format). Podprti so tudi različni vsebovalniki, kot je OVF (*angl.* Open Virtualization Format), vendar zaenkrat ostale komponente uporabljajo slike brez vsebovalnikov [68].

3.4.2 Nova

Nova je osrednja komponenta za izstavitev, gostovanje in upravljanje računskih virov oblačne infrastrukture z namenom nudenja infrastrukture kot storitve. Zasnovana je za enostavno horizontalno povečevanje zmogljivosti in je kompatibilna z različnimi virtualizacijskimi tehnologijami oziroma hipervizorji, s pomočjo katerih uporabnikom dodeljuje računske vire v obliki virtualnih strojev ali instanc iz bazena virov.

Računska storitev Nova je sestavljena iz več procesov, ki lahko tečejo na gruči strežnikov in med seboj komunicirajo s pomočjo oddaljenih klicev metod in sporočilnih vrst po protokolu AMQP (*angl.* Advanced Message Queuing Protocol). Privzeto se za ta namen uporablja sporočilni sistem

RabbitMQ, podprta pa sta tudi ZeroMQ in Qpid. Z ostalimi komponentami Nova komunicira preko aplikacijskih programskih vmesnikov REST. Za avtentikacijo tako uporablja storitev Keystone, slike navideznih diskov za ustvarjanje novih instanc pridobiva iz storitve Glance, za delo z omrežji si pomaga z omrežno storitvijo Neutron, trajno shrambo podatkov pa lahko instancam priključi s pomočjo bločne shrambe Cinder. Za shranjevanje lastnih podatkov o računskih virih uporablja relacijsko podatkovno bazo, ki si jo običajno deli z ostalimi storitvami [4, 68]. Ključni procesi računske storitve so:

- **Nova-api** - Zagotavlja aplikacijski programski vmesnik za končne uporabnike računske storitve in poseben administratorski programski vmesnik za privilegirane uporabnike. Uveljavlja nekatere politike in izvaja orkestracijo aktivnosti potrebnih za celoten življenjski cikel instanc.

- **Nova-api-metadata** - Instancam omogoča dostop do specifičnih podatkov o instanci npr. imena instance in ključev SSH (*angl.* Secure Shell) za dostop, ki jih lahko instanca uporabi za avtomatsko konfiguracijo s pomočjo skripte Cloud-init ob prvem zagonu. Dostop do podatkov iz instance je možen bodisi preko omrežja, tako da instanca pošlje zahtevo HTTP na naslov 169.254.169.254, ki jo omrežna storitev posreduje storitvi metapodatkov, ali v obliki konfiguracijskega diska, ki ga Nova priključi instanci ob kreiranju. Vmesnik je združljiv z Amazonovo storitvijo EC2 metadata.

- **Nova-scheduler** - V procesu ustvarjanja instance na podlagi filtrov in uteži določi ciljno računsko vozlišče, ki mu preko sporočilne vrste posreduje zahtevo za izstavitev nove instance.

- **Nova-conductor** - Služi kot posrednik med procesom nova-compute in podatkovno bazo. Računska vozlišča tako nimajo direktnega dostopa do podatkovne baze, kar je dobro iz varnostnega vidika in poenostavi nadgradnje [69].

- **Nova-compute** - Ustvarja, terminira in upravlja instance preko programskega vmesnika hipervizorja na računskem vozlišču ter posodablja trenutno stanje instanc v bazi. Podpira različne hipervizorje med drugim KVM preko libvirt, VMware vSphere, XenServer in HyperV.

- **Nova-novncproxy** - Deluje kot posrednik pri konzolnem dostopu do instanc iz brskalnika s pomočjo noVNC HTML5 odjemalca. Na zahtevo po konzolnem dostopu validira žeton za dostop s pomočjo procesa nova-consoleauth, vzpostavi povezavo do VNC strežnika na ustreznem hipervizorju in služi kot posredovalni strežnik za čas seje.

- **Nova-consoleauth** - Skrbi za avtentikacijo uporabniških žetonov pri konzolnem dostopu ter na podlagi žetona razreši naslov in vrata VNC strežnika za željeno instanco.

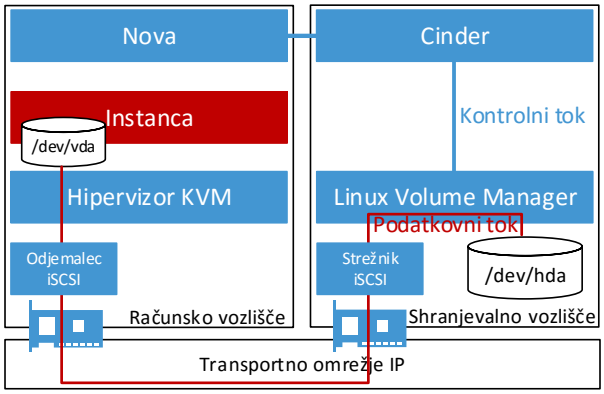
- **Nova-cert** - Je proces za upravljanje s certifikati po standardu x509, ki so potrebni pri uporabi EC2 API.

3.4.3 Cinder

Cinder zagotavlja storitev trajne bločne shrambe v obliki bločnih enot oziroma navideznih diskov, ki jih lahko pripnemo instancam ali pa instance iz njih celo zaganjamo. Privzeto so instance ustvarjene zgolj z začasnim navideznim diskom iz katerega se zaženejo, vse spremembe pa se izgubijo s terminacijo instance. Za trajno shranjevanje podatkov tako potrebujemo bločne enote, ki so neodvisne od življenjskega cikla instance [5].

Cinder ponuja gonilnike za različne tipe shramb. Privzeto se uporablja LVM, kar pomeni, da so navidezni diski ustvarjeni na lokalnih diskih shranjevalnih vozlišč s pomočjo odprtokodne programske opreme LVM, kot prikazuje slika 3.3. Na shranjevalnih vozliščih teče tudi strežnik iSCSI, ki logični disk deli preko omrežja IP z računskim vozliščem, kjer se nahaja instanca. Hipervizor nato poskrbi, da je navidezni disk viden instanci kot lokalna bločna enota. Takšen način shranjevanja zaenkrat ne podpira redundance za primer

odpovedi shranjevalnega vozlišča, zato moramo za redundanco podatkov poskrbeti na nivoju instance. Če želimo visoko razpoložljivo shrambo za navidezne diske, lahko uporabimo Ceph. Gre za odprtokodno programsko opremo, ki omogoča izredno skalabilno distribuirano shrambo na množici običajnih strežnikov. Bločne enote so razbite v množico objektov, ki so skupaj z redundantnimi kopijami razpršeni med vozlišči, pri čemer sistem poskrbi za avtomatsko zaznavanje in odpravljanje napak ter prerazporejanje podatkov med vozlišči [70]. Poleg omenjenih programskih rešitev so na voljo tudi gonilniki za namenske shranjevalne sisteme večjih proizvajalcev, kot so EMC, PureStorage, NetApp, HP in drugi. To omogoča tesno integracijo sistemov SAN in NAS v storitev bločne shrambe [71].



Slika 3.3: Bločna shramba z gonilnikom LVM [5].

Ne glede na uporabljeno shrambo oziroma njihovo kombinacijo Cinder zagotavlja abstrakcijo shrambe, saj skrije podrobnosti shranjevalnih sistemov in omogoči delo z navideznimi diski preko enotnega programskega vmesnika. Komponenta je sestavljena iz več procesov [4]:

- **Cinder-api** - Zagotavlja aplikacijski programski vmesnik za storitev bločne shrambe.
- **Cinder-scheduler** - Optimalno razporeja zahteve glede na zahtevano kapaciteto, tip shrambe, območje razpoložljivosti in druge parametre.

- **Cinder-volume** - Služi kot operativni vsebovalnik za gonilnike shrambe, ki omogočajo delo z različnimi shranjevalnimi sistemi.

- **Cinder-backup** - Omogoča storitev varnostnega kopiranja navideznih diskov v objektno shrambo OpenStack Swift ali drug tip shrambe.

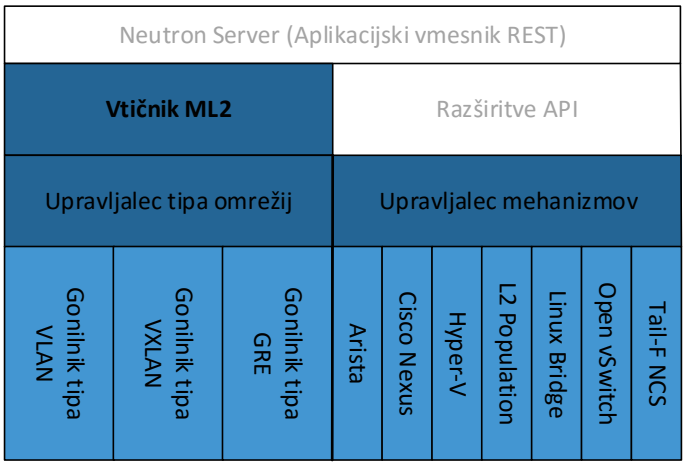
3.4.4 Neutron

Neutron je omrežna komponenta, ki s pomočjo vtičnikov zagotavlja razširljiv in skalabilen vmesnik za izstavitev ter upravljanje navidezne omrežne infrastrukture v večnajemniškem okolju. Uporabnikom na zahtevo omogoča enostavno ustvarjanje navideznih omrežij oziroma segmentov na drugi omrežni plasti, povezovanje instanc v omrežja in definiranje naslovnega prostora IP. Zasebni naslovi IP se lahko pri tem med uporabniki prekrivajo. Poleg samih omrežij Neutron omogoča tudi izstavitev usmerjevalnikov in drugih omrežnih storitev, kot so delilniki bremena ali požarne pregrade. Pomemben je koncept varnostnih skupin, o katerih smo govorili že v poglavju oblačne infrastrukture. Skupine določajo pravila dostopa, instance pa lahko pripadajo eni ali več skupin, na podlagi česar so določena dejanska pravila za komunikacijo na požarnih pregradah. Med drugim lahko instancam pripnemo tako imenovane plavajoče naslove IP. Gre za javne naslove IP, ki se s pomočjo preslikave omrežnih naslovov (*angl.* Network Address Translation - NAT) na usmerjevalniku preslikajo v zasebni naslov instance. Tako lahko omogočimo zunanji dostop do instance, kadar ta ni direktno priključena v omrežje z javnimi naslovi.

Pri upravljanju vseh omrežnih storitev Neutron skrije dejansko implementacijo, ki je popolnoma odvisna od uporabljenega vtičnika. Proces neutron-server zagotavlja aplikacijski programski vmesnik za omrežno storitev in posreduje zahteve ustreznim vtičnikom oziroma njihovim agentom. Neutron podpira kopico jedrnih vtičnikov, ki implementirajo različne programske rešitve ali pa omogočajo integracijo fizičnih omrežnih naprav v omrežno storitev, med drugim Open vSwitch, Linux Bridge, VMware NSX, Brocade

in Cisco Nexus.

Običajne monolitne vtičnike počasi nadomešča modularni vtičnik druge omrežne plasti ML2. Arhitekturo vtičnika lahko vidimo na sliki 3.4. Gre za ogrodje, ki omogoča uporabo različnih omrežnih tehnologij istočasno. Na voljo je več tipov omrežij, kot so VLAN, VXLAN in GRE, ter različni mehanizmi dostopa do teh omrežij realizirani s pomočjo gonilnikov. Trenutno so na voljo gonilniki za Open vSwitch, Linux Bridge, Tail-f NCS, Cisco APIC, Hyper-V agenta, OpenDaylight in še nekaj drugih. Sčasoma naj bi tudi ostale vtičnike prepisali v obliko gonilnikov mehanizmov s skupno kodo v ML2 in monolitne opustili.



Slika 3.4: Arhitektura vtičnika ML2 [6].

Neutron - modularni vtičnik ML2 in Open vSwitch

Pogledali si bomo, kako lahko omrežno storitev realiziramo z uporabo modularnega vtičnika ML2 v kombinaciji z navideznim stikalom Open vSwitch. V nadaljevanju bomo to kombinacijo uporabili pri vzpostavitvi infrastrukture zasebnega oblaka, ki bo vsebovala navidezna prekrivna omrežja z enkapsulacijo GRE. V tem primeru omrežno storitev poleg procesa neutron-server sestavljajo:

- **Neutron-plugin-ml2** - To je servisni proces za modularni vtičnik druge omrežne plasti ML2.

- **Neutron-plugin-openvswitch-agent** - Skrbi za konfiguracijo lokalnih navideznih stikal Open vSwitch na računskih in omrežnih vozliščih s pomočjo orodij ovs-ofctl in ovs-vsctl [72].

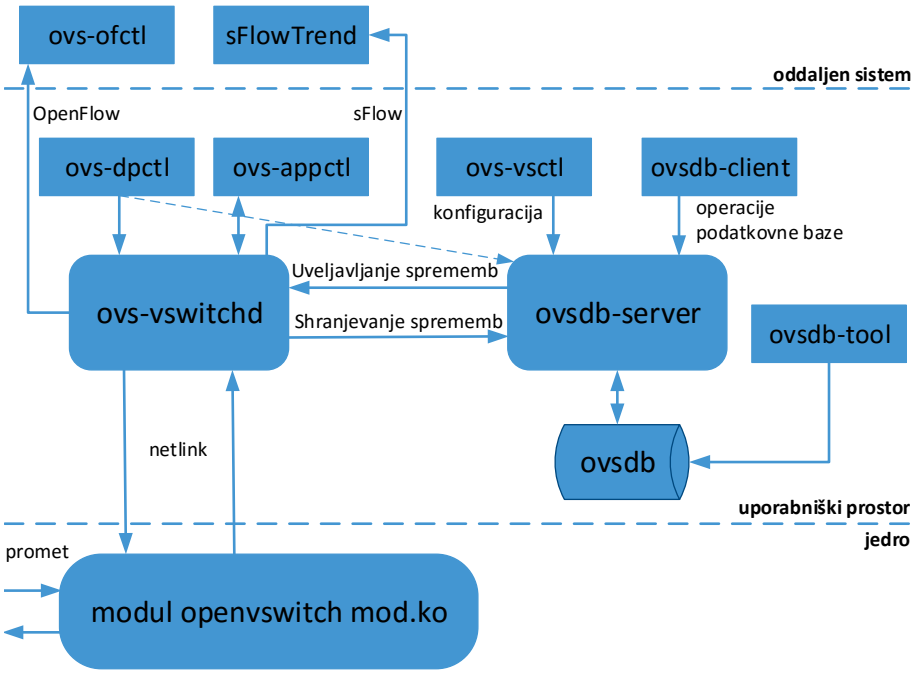
- **Neutron-l3-agent** - Zagotavlja storitev usmerjanja paketov na tretji omrežni plasti in preslikave omrežnih naslovov s pomočjo omrežnega sklada v operacijskem sistemu Linux ter programske opreme iptables. Posamezni navidezni usmerjevalniki so izstavljeni v obliki omrežnega imenskega prostora (*angl.* Network namespace), ki omogoča ločevanje omrežnih kontekstov na nivoju jedra operacijskega sistema. Omrežni imenski prostori imajo popolnoma izolirane omrežne vmesnike, sklade protokola IP, usmerjevalne tabele in imenik /proc/net. Z uporabo navideznega etherneta (*angl.* Veth) so imenski prostori nato povezani s primarnim imenskim prostorom sistema, kjer se nahaja navidezno stikalo Open vSwitch in fizične omrežne naprave.

- **Neutron-dhcp-agent** - Zagotavlja storitev strežnika DHCP (*angl.* Dynamic Host Configuration Protocol) za navidezna omrežja. Posamezne instance strežnika DHCP so ravno tako realizirane v ločenih omrežnih imenskih prostorih, kjer teče programska oprema Dnsmasq.

- **Neutron-metadata-proxy** - Teče v omrežnem imenskem prostoru usmerjevalnika, kjer s pomočjo pravila v iptables prestreza zahteve HTTP, ki prihajajo na naslov IP 169.254.169.254 in vrata 80. To so zahteve po metapodatkih instance. Proces v glavo HTTP doda izvorni naslov IP (naslov instance) ter ID usmerjevalnika in posreduje zahtevo procesu neutron-metadata-agent, ki teče v primarnem imenskem prostoru. Ko prejme odgovor, ga vrne ustrezni instanci [73].

- **Neutron-metadata-agent** - Za zahteve, ki jih prejme od neutron-metadata-proxy, na podlagi ID usmerjevalnika in naslova IP identifi-

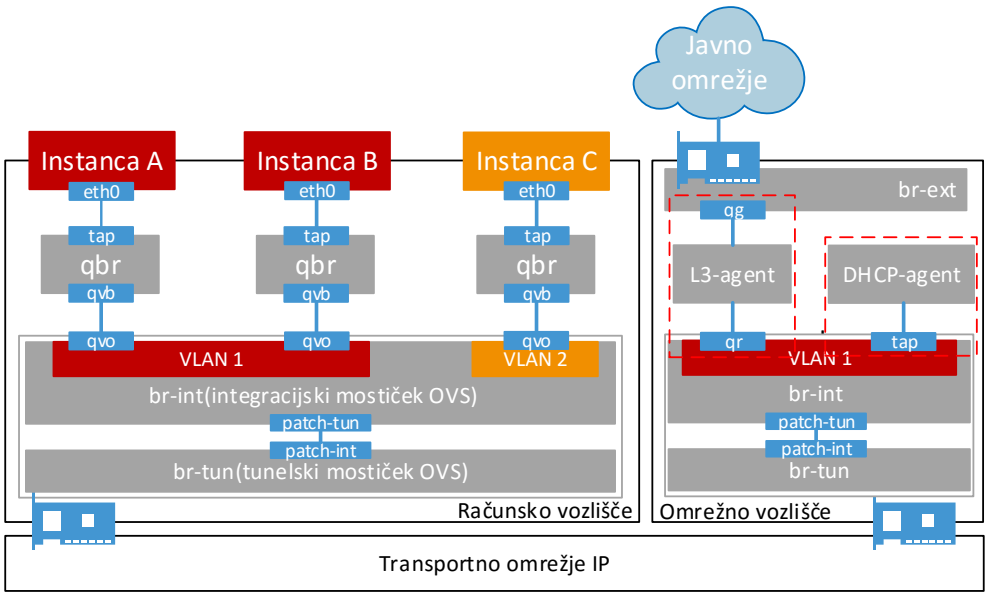
cira instanco. V glavo zahteve doda ID instance ter podpis za avtentikacijo zahteve. Zahtevo posreduje storitvi metapodatkov (nova-api-metadata) in odgovor posreduje nazaj instanci preko posrednika neutron-metadata-proxy [73].



Slika 3.5: Arhitektura stikala Open vSwitch [7].

Navidezno stikalo Open vSwitch je realizirano z demonom ovs-vswitchd, ki za hitro posredovanje okvirjev in enkapsulacijo prometa uporablja modul openvswitch_mod.ko jedra operacijskega sistema. Shranjevanje konfiguracije in pravil omogoča ovsdb-server, ki vzdržuje bazo ovsdb [7]. Arhitekturo stikala prikazuje slika 3.5.

Na sliki 3.6 lahko vidimo, kako so posamezne instance na računskem vozlišču povezane na navidezno stikalo Open vSwitch. Hipervizor na računskem vozlišču instanci zagotovi mrežni vmesnik eth0, ki povezuje instanco z navideznim vmesnikom ethernet tap na gostitelju. Vmesnik tap je pripet Linux mostičku qbr, kjer so s pomočjo iptables realizirane varnostne skupine ozi-



Slika 3.6: Open vSwitch na računskem vozlišču in omrežnem vozlišču.

roma ustrezna varnostna pravila. Navidezni par vmesnikov ethernet qvb in qvo nato povezuje mostiček qbr (varnostne skupine) z mostičkom br-int navideznega stikala Open vSwitch. To je integracijski mostiček, ki deluje kot običajno stikalo na drugem omrežnem nivoju. Izvaja označevanje prometa z oznako omrežja VLAN glede na vmesnik, preko katerega prihaja, oziroma posredovanje prometa ustreznim vmesnikom na podlagi oznake. Omogoča komunikacijo med direktno priključenimi instancami, ki so v istem navideznem omrežju.

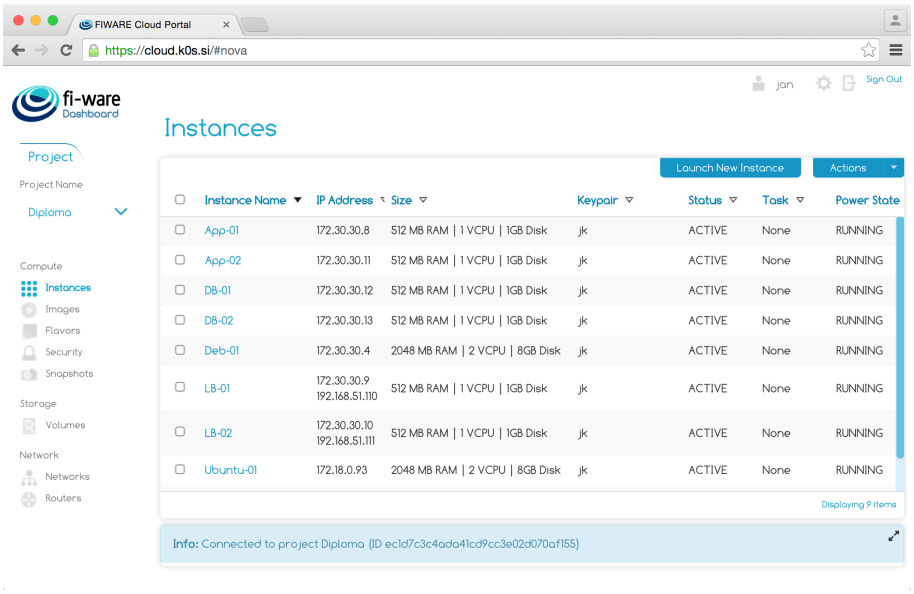
Vmesnik patch-tun povezuje integracijski mostiček s tunelskim mostičkom br-tun, kjer se izvaja enkapsulacija GRE za komunikacijo z instancami ali storitvami na drugih vozliščih. Ko se vozlišče priključi omrežni storitvi, agent poskrbi za vzpostavitev tunelov GRE z vsemi preostalimi vozlišči. Tuneli omogočajo transport okvirjev preko omrežja IP. Vmesniki tunelov GRE so priključeni na br-tun, ID tunelov pa se uporablja za ločevanje navideznih omrežij pri enkapsulaciji. Pravila OpenFlow na br-tun določajo translacijo med oznakami omrežja VLAN in ID tunela ter pravila za posredovanje preko

ustreznih tunelov oziroma na integracijski mostiček.

Na omrežnem vozlišču so na navidezno stikalo povezani imenski prostori, kjer so realizirani navidezni usmerjevalniki in strežniki DHCP, podobno kot so na računskem vozlišču povezane instance. Imenski prostori, kjer so realizirani usmerjevalniki, so na eni strani z vmesniki qr povezani v navidezna omrežja stanovalcev preko integracijskega mostička. Označevanje in posredovanje okvirjev preko tunelskega mostička poteka enako kot na računskih vozliščih. Na drugi strani so imenski prostori usmerjevalnikov preko vmesnika qg povezani na mostiček br-ext, ki jih povezuje z javnim zunanjim omrežjem [67, 74, 75, 76].

3.5 Samopostrežni portal - Cloud Portal

Samopostrežni portal Cloud Portal uporabnikom omogoča enostavno upravljanje virov oziroma storitev oblačne infrastrukture v obliki grafičnega spletnega vmesnika, ki z ostalimi komponentami komunicira preko programskega vmesnika REST [63]. V osnovi ponuja naprednejši vmesnik za funkcionalnosti, ki jih sicer nudi komponenta OpenStack Horizon, torej izstavitev, brisanje in urejanje navideznih virov, kot so instance, slike navideznih diskov, bločne enote, omrežja in varnostne skupine. Osnovni pogled na vmesnik lahko vidimo na sliki 3.7. Poleg upravljanja virov infrastrukture omogoča tudi delo z drugimi gradniki platforme FIWARE in storitvami na višjih nivojih. Tako npr. omogoča delo s predlogami pri nudenju platforme kot storitve s komponento Pegasus PaaS Manager [77].



Slika 3.7: Samopostrežni portal FIWARE Cloud Portal.

Grafični vmesnik je realiziran v obliki spletne aplikacije HTML5 na ogrodju Backbone.js, ki sledi principu Model-Pogled-Kontroler (*angl.* Model–View–Controller). Podatke izmenjuje s strežnikom Node.js, ki komunicira z ostalimi komponentami infrastrukture, uporabniški vmesnik pa se dinamično izrisuje

na odjemalski strani v brskalniku. Na strežniški strani potrebuje administratorski dostop do storitve Keystone. Za boljšo uporabniško izkušnjo ponuja odziven dizajn, ki se prilagaja napravi iz katere dostopamo. Ravno tako je vmesnik na voljo v več jezikih in omogoča enostavno dodajanje prevodov [78].

Poglavje 4

Vzpostavitev zasebnega oblaka FIWARE

Pogledali si bomo praktičen primer integracije posameznih komponent, ki smo jih podrobneje spoznali v prejšnjem poglavju, v delujočo oblačno infrastrukturo zasebnega oblaka za nudenje infrastrukture kot storitve. Infrastrukturo bomo preizkusili tako z uporabniškega kot administratorskega vidika ter analizirali njeno delovanje.

V ta namen smo implementirali demonstracijsko okolje s pomočjo vgnezdene virtualizacije (*angl.* Nested virtualization) na strežniku VMware ESXi. Računsko storitev smo realizirali z uporabo hipervizorja KVM. Za realizacijo omrežne storitve smo uporabili kombinacijo modularnega vtičnika ML2, navideznega stikala Open vSwitch in navideznih prekrivnih omrežij z enkapsulacijo GRE. Za trajno shrambo podatkov smo s storitvijo bločne shrambe in shrambe slik navideznih diskov integrirali omrežno dostopno shrambo Synology. Vključili smo tudi vse podporne storitve, ki so potrebne za delovanje, kakor tudi samopostrežni portal. Za vozlišča oblačne infrastrukture smo izbrali operacijski sistem Linux Ubuntu 14.04 LTS, saj je dobro podprt v okviru oblačne platforme OpenStack, na kateri temelji FIWARE, različica 14.04 LTS pa je bila v času postavitve aktualna izdaja s podaljšano podporo. Kot podatkovno bazo smo namestili odprtokodno programsko opremo

MariaDB in za sporočilni sistem uporabili RabbitMQ.

4.1 Arhitektura

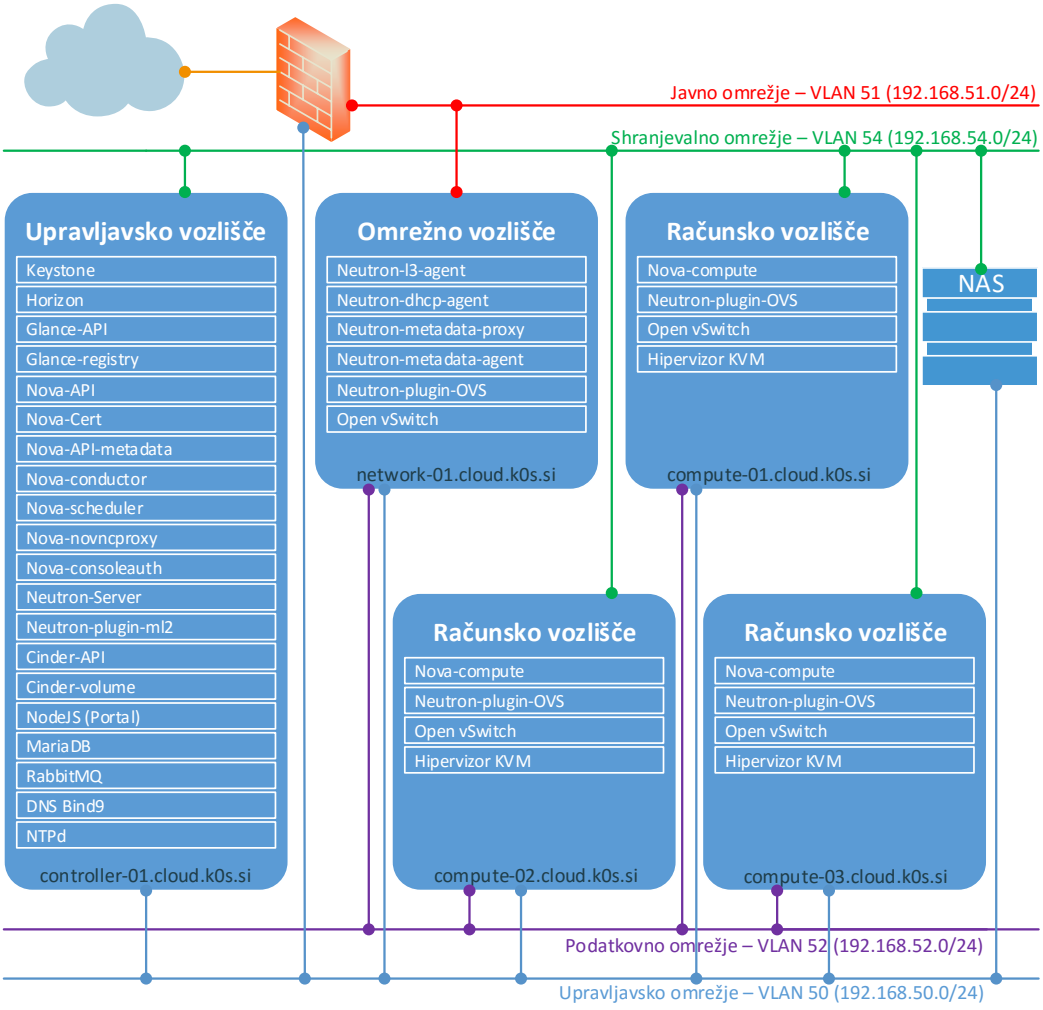
Začeli smo s pripravo načrta ciljne arhitekture, ki bo služil kot izhodišče pri implementaciji oblaka. Glede na zasnovo komponent, ki smo jih spoznali v 3. poglavju, smo se odločili za postavitev okolja na petih vozliščih:

- **Upravljavsko vozlišče** - Namenjeno je vsem upravljavskim storitvam oblačne infrastrukture. Sem spada celotna storitev upravljanja z identitetami ter upravljavske komponente ostalih storitev, in sicer: Glance-API, Glance-registry, Nova-API, Nova-Cert, Nova-api-metadata, Nova-conductor, Nova-scheduler, Nova-novncproxy, Nova-consoleauth, Neutron-Server, Neutron-plugin-ml2, Cinder-API, Cinder-Volume in samopostrežni portal Cloud Portal. Na upravljavskem vozlišču se nahajajo tudi podporne storitve, ki so skupne vsem komponentam. To je strežnik DNS Bind9, strežnik NTP, relacijska podatkovna baza MariaDB in sporočilni sistem RabbitMQ.

- **Omrežno vozlišče** - Namenjeno je realizaciji omrežnih storitev, konkretno navideznih usmerjevalnikov (neutron-l3-agent), s katerimi povezuje navidezna omrežja z zunanjim svetom, strežnikov DHCP (neutron-dhcp-agent) in posredovanju metapodatkov instancam (neutron-metadata-proxy ter neutron-metadata-agent). Posamezne storitve so povezane z navideznim stikalom Open vSwitch, za ustrezno konfiguracijo stikala pa je nameščen neutron-plugin-openvswitch-agent.

- **3 računska vozlišča** - Vozlišča so namenjena izvajanju delovnih obremenitev oziroma poganjanju instanc. Na njih teče hipervizor KVM in proces Nova-compute, ki z njim upravlja. Na računskih vozliščih je tudi del omrežne storitve, in sicer stikalo Open vSwitch ter agent za konfiguracijo stikala neutron-plugin-openvswitch-agent.

Komponente za upravljanje so tako ločene od komponent za izvajanje delovnih obremenitev in omrežnih komponent. Računske in omrežne kapacitete je mogoče enostavno razširiti z dodajanjem novih vozlišč. Zmogljivosti upravljavskega dela je mogoče ravno tako horizontalno razširiti z uporabo delilnikov bremena in dodajanja novih vozlišč. Načrt ciljne arhitekture lahko vidimo na sliki 4.1.



Slika 4.1: Načrt arhitekture.

Fizično omrežje smo z uporabo navideznih lokalnih omrežij ločili na upravljavsko podomrežje, preko katerega bodo komunicirale posamezne kompo-

nente infrastrukture, podatkovno omrežje za prenos enkapsuliranega prometa navideznih prekrivnih omrežij, shranjevalno omrežje za povezavo s sistemom NAS in javno omrežje, ki je namenjeno povezavi navideznih usmerjevalnikov z zunanjim svetom. Iz nabora naslovov v javnem omrežju se dodeljujejo tudi plavajoči naslovi, ki jih lahko pripnemo posameznim instancam in se na navideznem usmerjevalniku preslikajo v zasebni naslov instance. Za podatkovno omrežje je pomembno, da zaradi dodatne enkapsulacije navideznih prekrivnih omrežjih podpira velikost okvirjev Ethernet oziroma vrednost MTU večjo od standardnih 1500 B. Za podatkovno omrežje smo zato predvideli uporabo tako imenovanih jumbo okvirjev.

4.2 Priprava okolja

Na podlagi načrta smo pripravili ustrezno okolje. Vozlišča predstavljajo fizične strežnike v oblačni infrastrukturi, vendar smo za potrebe tega okolja uporabili navidezne stroje na hipervizorju VMware ESXi. Poskrbeli smo za konfiguracijo navideznih lokalnih omrežij na stikalu in pripravili 5 navideznih strežnikov z operacijskim sistemom Linux Ubuntu 14.04 LTS. Hipervizor KVM v našem primeru teče v virtualnem stroju, zato smo strežnikom predhodno omogočili podporo za vgnezdeno virtualizacijo, ki ukaze za strojno virtualizacijo izpostavi navideznim strojem. Poskrbeli smo za osnovno konfiguracijo vozlišč z omrežnimi nastavitvami, imeni vozlišč in ključi SSH za oddaljen dostop. Upravljavsko in javno omrežje smo povezali na robni usmerjevalnik in požarno pregrado pfSense, ki nam omogoča povezavo s svetovnim spletom in oddaljeno administracijo preko navideznega zasebnega omrežja.

Komponente za dostop do baze, sporočilnega sistema in medsebojno komunikacijo uporabljajo različna uporabniška imena in gesla. Vsaka od komponent ima tako uporabniško ime in geslo za dostop do relacijske podatkovne baze MariaDB ter servisnega uporabnika v storitvi Keystone z ločenim uporabniškim imenom in geslom za dostop do aplikacijskih programskih vmesnikov preostalih komponent. Z geslom je zaščiten tudi dostop do sporočilnega

sistema RabbitMQ. Vsa gesla smo naključno generirali z orodjem openssl in ustrezno dokumentirali za poznejšo uporabo.

4.3 Namestitev in konfiguracija komponent

Sledila je namestitev in konfiguracija posameznih komponent na vozliščih. Pri namestitvi in konfiguraciji smo si primarno pomagali s prosto dostopno dokumentacijo za namestitev FIWARE DCRM [36] in komponent platforme OpenStack [4, 67]. V pomoč so nam bile tudi uradne dokumentacije različne odprtokodne programske opreme, ki je integrirana v platformo FIWARE.

4.3.1 Podporne storitve

Najprej smo namestili podporne storitve, ki jih ostale komponente potrebujejo za svoje delovanje. Za sinhronizacijo časa s protokolom NTP smo na vsa vozlišča namestili programsko opremo NTPd. Na upravljavskem vozlišču smo nastavili sinhronizacijo na Arnesova javna strežnika za točen čas. Vozlišče deluje kot strežnik NTP, na katerega se sinhronizirajo vsa ostala vozlišča. Točen čas je pomemben za časovno usklajeno delovanje aplikacij v omrežju zlasti pri mehanizmih za avtentikacijo.

Na upravljavsko vozlišče smo namestili bind9, ki služi kot interni strežnik DNS. Strežnik smo nastavili kot primarni avtoritativni strežnik za cono *cloud.k0s.si*, kamor smo dodali zapise za imena vozlišč infrastrukture in pripadajočo cono reverznih preslikav za upravljavsko omrežje. Omogočili smo posredovanje ostalih poizvedb na Googlova javna imenska strežnika.

Relacijsko podatkovno bazo MariaDB smo namestili iz repozitorijev, ki so vključeni v operacijski sistem. Nastavili smo naslov na katerem posluša strežnik, spominski stroj InnoDB in kodni nabor UTF8. Določili smo administratorskega uporabnika z geslom, ki smo ga predhodno pripravili.

Pri namestitvi in testiranju RabbitMQ smo imeli težave pri komunikaciji določenih komponent s sporočilnim sistemom. V privzetih repozitorijih apt je nekoliko starejša različica, zato smo namestili zadnjo različico iz uradnega

repozitorija, s katero nismo imeli težav. Nastavili smo še geslo za dostop do sporočilnega sistema in omogočili vmesnik za administracijo. S tem smo pripravili vse potrebno za vključitev komponent platforme FIWARE.

4.3.2 Upravljanje z identitetami

Komponenta za upravljanje z identitetami KeyRock od verzije 2.0 v celoti temelji na komponenti Keystone in je na voljo v obliki izvorne kode v repozitoriju Git. Najprej smo namestili in preizkusili različico iz omenjenega repozitorija, ki vsebuje razširitev za OAuth2.0. Zaenkrat razširitve nismo potrebovali, zato smo za boljšo kompatibilnost z ostalimi komponentami uporabili stabilni različici Keystone ter Horizon iz OpenStack Juno.

Slika 4.2: Dodajanje storitve za upravljanje identitet v katalog storitev.

V relacijski podatkovni bazi MariaDB smo ustvarili bazo keystone in uporabnika, ki ima dostop do baze. Iz repozitorijev apt smo na upravljavskem vozlišču namestili storitev keystone in odjemalca python-keystoneclient. Pri namestitvi se ustvari uporabnik keystone pod katerim teče storitev. V konfiguracijo smo vnesli niz za povezavo z bazo, izbrali gonilnik za shranjevanje

žetonov v bazi in določili začasni žeton za uporabnika admin, s katerim se lahko avtenticiramo preden v sistemu dejansko ustvarimo uporabnike. Z orodjem keystone-manage smo pod uporabnikom keystone sprožili inicializacijo podatkovne baze in s pomočjo skripte upstart zagnali storitev.

Storitev je tako dosegljiva preko vmesnika REST, vendar jo moramo za pravilno delovanje vnesti v katalog storitev skupaj z vstopno točko programskega vmesnika. Z odjemalcem smo se s pomočjo začasnega žetona povezali s storitvijo keystone, kjer smo ustvarili administratorski stanovalski objekt in globalnega administratorja admin, ki ga v nadaljevanju uporabljamo za administracijo. Nato smo ustvarili servisni stanovalski objekt v katerem se nahajajo servisni uporabniki za ostale komponente. V katalog smo dodali storitev upravljanja z identitetami in vstopno točko njenega programskega vmesnika, kot lahko vidimo na sliki 4.2.

Namestitev grafičnega uporabniškega vmesnika za administracijo Horizon je enostavna. Namestili smo ga s pomočjo paketa openstack-dashboard, pri čemer potrebujemo še strežnik apache2, libapache2-mod-wsgi, memcached in python-memcache. Odstranili smo temo dashboard-ubuntu, ki je privzeto vključena v paket za Ubuntu. V konfiguracijsko datoteko smo vnesli naslov storitve Keystone, nastavili memchached za shranjevanje sejnih podatkov in pognali spletni strežnik.

4.3.3 Shranjevanje slik

Storitev za shranjevanje slik Glance smo v celoti namestili iz privzetega repozitorija apt. Z odjemalcem mysql smo ustvarili novo podatkovno bazo glance in uporabnika za dostop do baze. V storitvi keystone smo pripravili istoimenskega servisnega uporabnika in dodali storitev ter vstopno točko po istem postopku kot za samo storitev Keystone. V konfiguracijskih datotekah za glance-api in glance-registry smo podali niz za dostop do baze, vstopno točko storitve keystone, ime servisnega stanovalskega objekta ter geslo in uporabniško ime za dostop.

Slike navideznih diskov želimo hraniti na omrežno dostopni shrambi po-

datkov, zato smo na sistemu NAS ustvarili nov imenik imenovan images ter pripadajoč izvoz NFS. Glance nima specifičnega gonilnika za shrambo NAS, zato smo uporabili gonilnik za hrambo na datotečnem sistemu. Na upravljavsko vozlišče smo namestili paket nfs-common, ki vsebuje odjemalca za dostop do omrežnega datotečnega sistema. Z zapisom v /etc/fstab smo poskrbeli za avtomatski priklop izvoza 192.168.54.5:/volume3/images na lokalni imenik /var/lib/glance/images, kamor glance shranjuje slike navideznih diskov. Z ukazoma chown in chmod smo nastavili lastništvo ter pravice za uporabnika glance nad imenikom. Sledila je le še inicializacija baze z orodjem glance-manage in zagon storitev glance-api ter glance-registry.

4.3.4 Računska storitev

Za računsko storitev smo tako kot za ostale storitve najprej pripravili podatkovno bazo in dodali servisnega uporabnika, storitev ter vstopno točko za programski vmesnik v storitvi Keystone. Na upravljavsko vozlišče smo namestili pakete nova-api, nova-cert, nova-conductor, nova-consoleauth, nova-novncproxy in nova-scheduler, ki zajemajo upravljavski del storitve Nova, ter odjemalca python-novaclient. Poleg niza za dostop do baze in podatkov o storitvi Keystone smo v konfiguracijski datoteki tokrat podali še tip, naslov in geslo sporočilnega sistema, ki ga Nova potrebuje za komunikacijo med porazdeljenimi procesi. Del storitve je namreč na računskih vozliščih, kjer se izvajajo delovne obremenitve. Vnesti smo morali še naslov IP, na katerem posluša posredovalni strežnik za VNC, ki omogoča konzolni dostop do instanc. Po inicializaciji podatkovne baze z orodjem nova-manage, smo na računska vozlišča namestili paket nova-compute. Konfiguracija je enaka kot na upravljavskem vozlišču brez dostopnih podatkov za bazo, dodati smo morali le še naslov posredovalnega strežnika VNC in določiti tip hipervizorja. V našem primeru je to KVM, ki je del jedra operacijskega sistema Linux in ga ni potrebno posebej nameščati.

Nova privzeto ustvarja instance na lokalnem datotečnem sistemu računskih vozlišč v imeniku /var/lib/nova/instances, kamor prenese sliko navideznega

diska in iz nje ustvari začasni disk iz katerega se instanca zažene. Odločili smo se, da uporabimo deljeno shrambo, kar nam omogoča enostavno selitev instanc med vozlišči in tako poenostavi vzdrževanje. Na sistemu NAS smo pripravili imenik instances ter pripadajoč izvoz NFS. Na računskih vozliščih smo namestili paket nfs-common in z zapisom v /etc/fstab poskrbeli za avtomatski priklop izvoza 192.168.54.5:/volume1/instances na lokalni imenik /var/lib/nova/instances. Vsa računska vozlišča tako vidijo enoten imenik z instancami. Možnost selitve živih instanc je bilo potrebno omogočiti v konfiguraciji vmesnika libvirt, preko katerega nova-compute upravlja s hipervizorjem KVM. V konfiguraciji storitve Nova smo popravili še naslov strežnika VNC, ki ne sme vsebovati dejanskega naslova IP temveč naslov 0.0.0.0, saj se naslov vozlišča ob selitvi instance spremeni.

4.3.5 Omrežna storitev

Po enakem postopku smo pripravili podatkovno bazo, servisnega uporabnika in vpis v katalog storitev tudi za omrežno storitev. Na upravljavsko vozlišče smo namestili neutron-server, modularni vtičnik neutron-plugin-ml2 in odjemalca python-neutronclient. Sledile so nastavitve za dostop do baze, sporočilnega sistema in storitve za upravljanje z identitetami. V nastavitvah smo omogočili uporabo vtičnika ML2 ter storitev navideznih usmerjevalnikov in dovolili uporabo prekrivajočih se naslovov IP. V nastavitvah modularnega vtičnika ML2 smo vključili omrežja tipa flat za javno omrežje in GRE za omrežja stanovalcev, izbrali gonilnik za Open vSwitch, določili območje tunelskih identifikatorjev in omogočili varnostne skupine. V konfiguraciji storitve Nova smo specificirali uporabo omrežne storitve Neutron in z orodjem neutron-db-manage inicializirali podatkovno bazo za omrežno storitev.

Na računska vozlišča smo namestili modularni vtičnik neutron-plugin-ml2 in agenta neutron-plugin-openvswitch-agent. Konfiguracija je enaka kot na upravljavskem vozlišču brez podatkov za dostop do baze, saj omrežna in računska vozlišča nimajo direktnega dostopa. Dodati smo morali le tip tunelov in lokalni naslov IP, ki se uporablja za terminacijo tunelov GRE. V

našem primeru je to naslov računskega vozlišča v podatkovnem omrežju.

Na omrežnem vozlišču smo najprej omogočili posredovanje paketov s parametri jedra v datoteki /etc/sysctl.conf. Namestili smo modularni vtičnik neutron-plugin-ml2 in agente neutron-plugin-openvswitch-agent, neutron-l3-agent in neutron-dhcp-agent. Osnovna konfiguracija je enaka kot na računskih vozliščih, dodali smo le še konfiguracijo za javno omrežje ter agenta L3 in DHCP. Določili smo uporabo navideznega stikala OVS za L3 in DHCP, mostiček za povezavo z zunanjim svetom (br-ex) in uporabo ločenih imenskih prostorov, ki omogočajo uporabo prekrivajočih se naslovov IP. V konfiguraciji agenta za metapodatke smo določili deljeno skrivnost, ki se uporablja za podpis in avtentikacijo zahtev. Storitev metapodatkov smo morali na to omogočiti še na upravljavskem vozlišču in vnesti deljeno skrivnost, ki smo jo določili.

V omrežni storitvi smo nato z odjemalcem ustvarili omrežje ext-net. To je fizično javno omrežje naše infrastrukture, ki je preko mostička br-ex dostopno vsem stanovalcem za povezavo navideznih usmerjevalnikov z zunanjim svetom. Omrežju smo pripeli javni omrežni segment, kjer smo določili območje naslovov IP, ki se uporabljajo za navidezne usmerjevalnike in plavajoče naslove ter naslov IP privzetega prehoda.

4.3.6 Bločna shramba

Za storitev bločne shrambe smo pripravili podatkovno bazo imenovano cinder in pripadajočega uporabnika za dostop do baze. V storitvi Keystone smo ustvarili servisnega uporabnika. Tokrat smo dodali dve storitvi, in sicer cinder ter cinderv2 za obe verziji programskih vmesnikov, katerih vstopne točke smo ravno tako dodali v katalog storitev. Na upravljavsko vozlišče smo nato iz privzetega repozitorija apt namestili pakete cinder-api, cinder-scheduler in cinder-volume. V konfiguraciji smo podali podatke za dostop do podatkovne baze, sporočilnega sistema in storitve keystone ter inicializirali podatkovno bazo z orodjem cinder-manage.

Bločne enote oziroma navidezne diske smo želeli shranjevati na sistemu

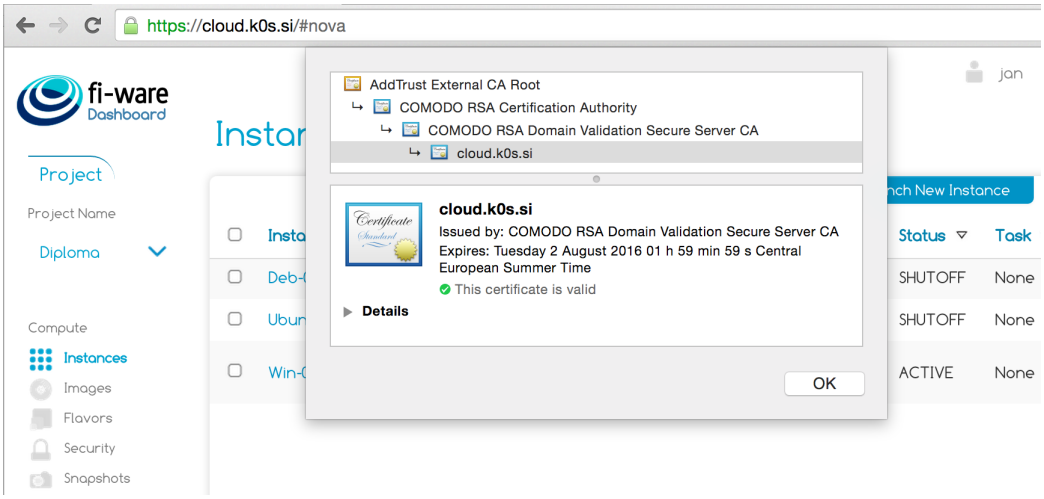
NAS, zato smo pripravili nov imenik volumes in izvoz NFS za ta imenik. Storitev cinder ima namenski gonilnik za NFS, ki poskrbi za pripenjanje omrežnega imenika na vozliščih, ki potrebujejo dostop. V konfiguracijo smo dodali izbiro gonilnika za NFS in v datoteko /etc/cinder/nfs_shares seznam omrežnih imenikov, v našem primeru je to 192.168.54.5:/volume1/volumes. Pred uporabo smo imenik še ročno pripeli na upravljavsko vozlišče, da smo lahko nastavili ustrezno lastništvo in pravice nad imenikom. Paket nfs-common, ki vsebuje odjemalca za NFS, smo na upravljavsko vozlišče in računska vozlišča namestili že pri vzpostavitvi ostalih storitev. Bločna shramba je tako pripravljena.

4.3.7 Samopostrežni portal

Pri namestitvi samopostrežnega portala smo imeli sprva kar nekaj težav. Portal je namreč vsaj za zdaj na voljo zgolj v obliki izvorne kode, navodila za namestitev pa so bila nekoliko pomanjkljiva in namenjena predvsem vzpostavitvi razvojnega okolja. Namestitev zahteva strežnik NodeJS, prevajalnik G++, interpreter za programski jezik ruby in orodje Saas. Kodo smo prenesli iz uradnega repozitorija Git, vendar smo imeli težave s kompilacijo datotek CSS zaradi napačne verzije programske opreme Saas, kar smo uspeli rešiti z razvijalcem. V konfiguracijsko datoteko smo vnesli parametre za dostop do storitve Keystone. Določeni moduli se kljub temu niso ustrezno prikazovali v brskalniku. Ugotovili smo, da je težava v formatu vstopnih točk, ki jih portal pridobiva s pomočjo storitve Keystone. Portal namreč zahteva drugačno obliko, kot je bila navedena v uradni dokumentaciji za OpenStack. Ugotovili smo tudi, da so v programski kodi zapečeni določeni naslovi URL za testno okolje FIWARE Lab, zato nekatere funkcionalnosti, kot je konzolni dostop do instanc, niso delovale v našem okolju. Naslove smo prilagodili v skladu z našo postavitvijo. Razvijalci naj bi v kratkem prilagodili portal tako, da naslove strežnik prebere iz konfiguracijske datoteke, kjer so tudi ostali parametri.

Strežnik NodeJS v osnovi ne teče kot storitev. Preizkusili smo več načinov za zagon strežnika v obliki demona. Odločili smo se za orodje Forever[79],

ki omogoča pogon strežnika v ozadju, stalni nadzor procesa in enostavno delo z dnevniškimi datotekami. Ustvarili smo novega uporabnika pod katerim teče strežnik NodeJS in imenika za programsko kodo ter dnevniške datoteke po zgledu ostalih storitev. Pripravili smo skripto upstart, ki se nahaja v /etc/init/portal.conf in poskrbi za avtomatski zagon procesa NodeJS z orodjem Forever ob zagonu operacijskega sistema.

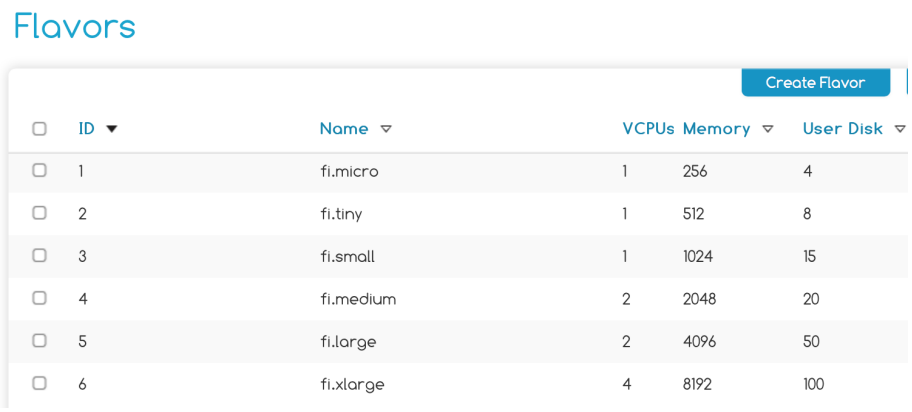


Slika 4.3: Digitalno potrdilo pri dostopu do portala.

Za varen dostop do samopostrežnega portala, smo omogočili samo dostop preko protokola HTTPS, ki za vzpostavitev varnega kanala uporablja kriptografski protokol SSL (*angl.* Secure Sockets Layer) oziroma TLS (*angl.* Transport Layer Security). Z orodjem openssl smo generirali zasebni ključ in zahtevo za podpis javnega ključa za naslov *cloud.k0s.si*, preko katerega je dostopen naš samopostrežni portal. Zahtevo za podpis smo posredovali certifikatni agenciji Comodo, ki nam je po preverbi lastništva domene izdala digitalno potrdilo. Na strežnik smo prenesli zasebni ključ in izdano digitalno potrdilo skupaj z digitalnimi potrdili vseh certifikatnih agencij v verigi zaupanja. V konfiguraciji strežnika smo podali pot do zasebnega ključa in digitalnih potrdil. Pri dostopu do portala lahko brskalnik tako preveri verodostojnost našega strežnika in vzpostavi šifriran kanal, kot vidimo na sliki 4.3.

4.4 Prikaz delovanja

Končni uporabniki infrastrukture kot storitve imajo dostop do samopostrežnega portala, kjer lahko upravljajo svoje vire. Kot upravitelj oblaka običajno v naprej pripravimo slike navideznih diskov, iz katerih uporabniki zaganjajo instance in definiramo tako imenovane okuse (*angl.* Flavors). Gre za tipe



Slika 4.4: Vnaprej definirani tipi instanc.

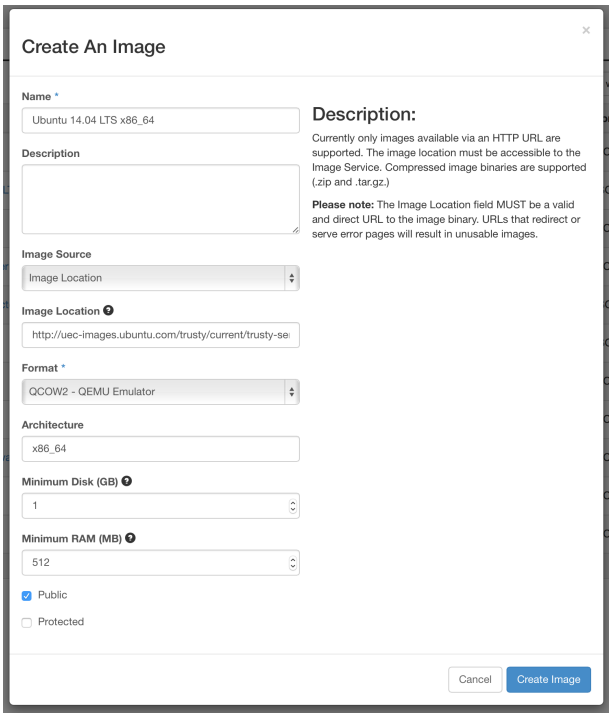
instanc oziroma velikostne skupine z vidika kapacitete pomnilnika, shrambe in navideznih procesorjev. Kot lahko vidimo na sliki 4.4, smo definirali 6 osnovnih tipov instanc, ki imajo od 1 do 4 navidezne procesorje, od 256MB do 8GB pomnilnika in od 4GB do 100GB prostora na začasnem disku.

4.4.1 Priprava slik

Slike navideznih diskov služijo kot predloge za zagon instanc in bistveno pospešijo izstavitev novih strežnikov. Vsebujejo že nameščen operacijski sistem, potrebne gonilnike in največkrat tudi orodja, ki omogočajo osnovno prilagoditev operacijskega sistema na podlagi metapodatkov. Orodje cloud-init nam tako ob zagonu omogoča avtomatsko nastavitev imena, omrežnih nastavitev, dodajanje dovoljenih ključev SSH za oddaljen dostop, razširitev particij glede na dejansko velikost navideznega diska, nastavljanje gesel in procesiranje kopice drugih metapodatkov [80]. Pri pripravi slik moramo upoštevati ciljno

arhitekturo in posebnosti hipervizorja, na katerem se bo instanca izvajala. Hipervizor sicer zagotavlja določen nivo abstrakcije strojne opreme, vendar kljub temu potrebujemo gonilnike za navidezne naprave. V našem primeru uporabe hipervizorja KVM so vhodno-izhodne naprave paravirtualizirane s pomočjo programske opreme qemu, zato instance za pravilno delovanje krmilnikov diska, mrežnih adapterjev in upravljanje pomnilnika potrebujejo gonilnike VirtIO. Ti so na voljo za vse moderne operacijske sisteme in jih lahko brezplačno prenesemo s spleta.

Za večino odprtokodnih operacijskih sistemov so na voljo že pripravljene slike navideznih diskov, ki vsebujejo gonilnike in orodje cloud-init. Slike



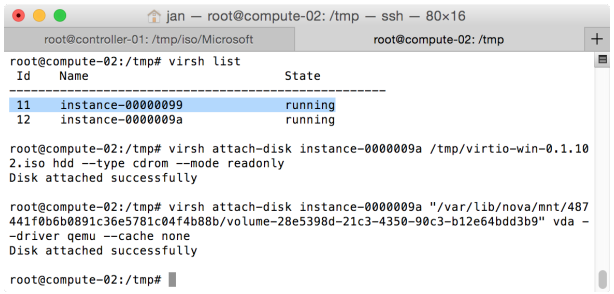
Slika 4.5: Uvoz slike navideznega diska Ubuntu LTS 14.04.

so kompatibilne z našo platformo kakor tudi z množico javnih oblakov, kot je Amazon EC2. Lahko jih enostavno prenesemo in uvozimo v shrambo slik navideznih diskov oziroma preko tekstovnega ali grafičnega vmesnika podamo naslov URL, iz katerega se v shrambo prenese slika, kot smo to

storili za operacijski sistem Ubuntu LTS 14.04 na sliki 4.5. Na enak način smo v shrambo dodali še slike za Debian, CentOS 7 in FreeBSD. V obliki slik navideznih diskov je na voljo tudi vedno več produktov. Tako smo dodali sliko s prednameščenim orodjem za avtomatizacijo oblačne infrastrukture ManageIQ in navidezni usmerjevalnik Cisco CSR1000V.

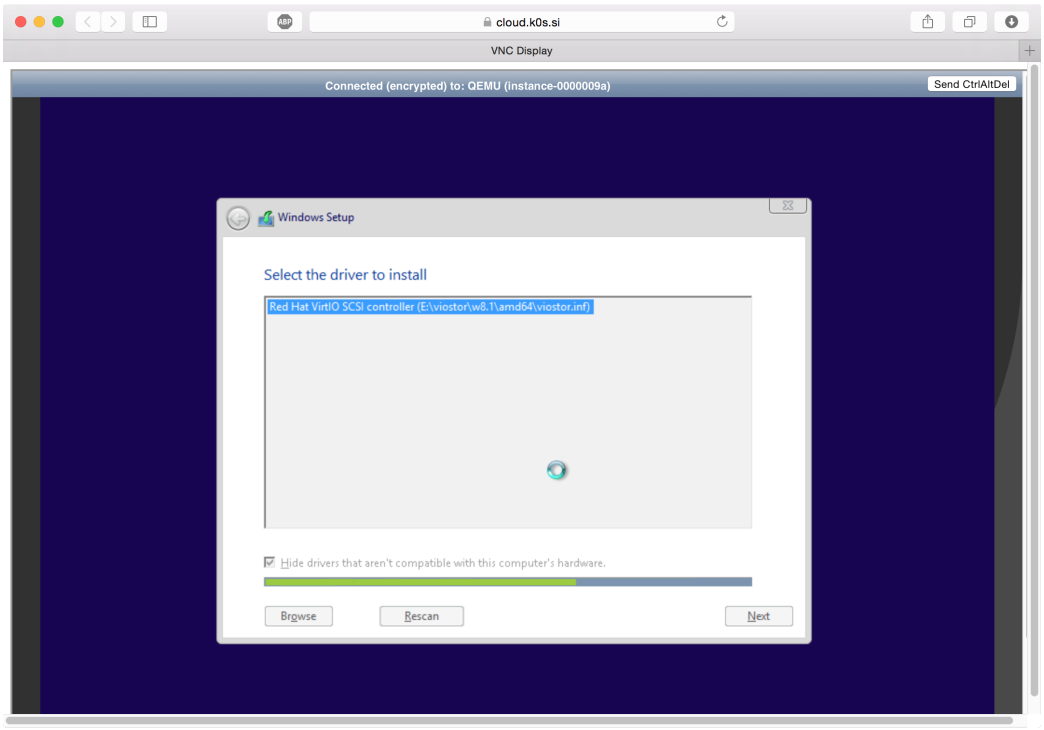
Za operacijski sistem Windows je na voljo orodje Cloudbase-Init, ki omogoča podobno prilagoditev operacijskega sistema kot cloud-init za ostale platforme. V okviru projekta Cloudbase je na voljo tudi predpripravljena slika z evalvacijsko različico operacijskega sistema Windows Server 2012 R2, ki jo lahko pozneje aktiviramo z ustrezno licenco.

Sami smo pripravili še sliko za operacijski sistem Windows 8.1, ki bi ga lahko uporabili za navidezna namizja. Sliko lahko pripravimo na poljubnem računalniku, vendar smo se odločili, da bomo za to uporabili kar našo oblačno infrastrukturo. Med slike lahko namreč uvozimo tudi namestitveni medij v formatu ISO, iz katerega zaženemo instanco. To smo tudi storili. Ustvarili smo novo bločno enoto za namestitev sistema, vendar smo naleteli na težavo. Ugotovili smo, da zaradi hrošča v trenutni verziji bločne shrambe, bločne enote ne moremo pripeti instanci, kadar instanco zaženemo iz slike ISO. Pomagali smo si tako, da smo se povezali neposredno na računsko vo-



Slika 4.6: Ročno pripenjanje bločne enote in slike ISO z gonilniki.

zlišče, kjer se je instanca zagnala in bločno enoto pripeli ročno (slika 4.6). Na enak način smo pripeli tudi sliko z gonilniki VirtIO, saj gonilnik za krmilnik diska potrebujemo že med namestitvijo (slika 4.7). Po uspešni namestitvi



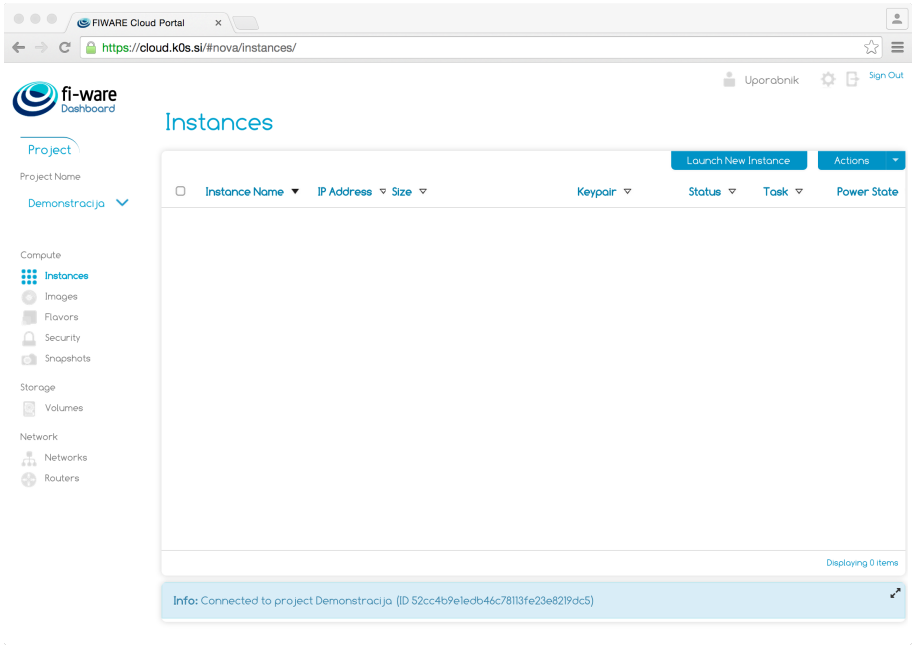
Slika 4.7: Dodajanje gonilnikov VirtIO pri namestitvi Windows 8.1.

smo zagnali novo instanco iz ustvarjene bločne enote. V instanco smo namestili še preostale gonilnike in orodje Cloudbase-init, instanco ugasnili ter posnetek bločne enote shranili v shrambo slik navideznih diskov. Iz te slike je sedaj mogoče zaganjati nove instance z operacijskim sistemom Windows 8.1, ki se ob zagonu ustrezno prilagodi. Podobno smo pripravili še sliko za operacijski sistem Linux Ubuntu Desktop 14.04. Distribucija Ubuntu že vsebuje gonilnike VirtIO, zato je bila priprava slike nekoliko enostavnejša.

4.4.2 Izstavitev virov

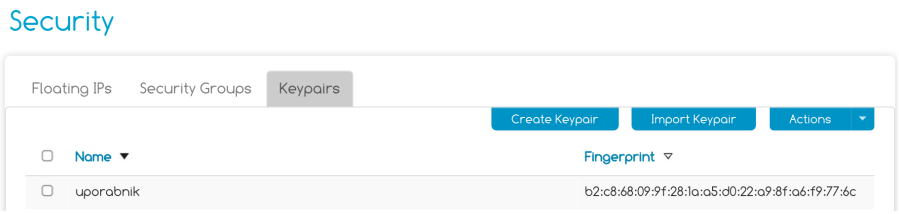
Naša oblačna infrastruktura je pripravljena za uporabo, zato si bomo ogledali, kako je izstavljanje virov videti iz uporabniškega vidika. Pripravili smo nov stanovalski objekt in uporabnika v tem stanovalskem objektu. Ob vstopu na samopostrežni portal se mora uporabnik najprej prijaviti z uporabniškim

imenom in geslom in nato lahko začne z izstavljanjem virov preko vmesnika, ki ga vidimo na sliki 4.8. Pogledali si bomo primer izstavitve instance, ki



Slika 4.8: Vmesnik samopostrežnega portala.

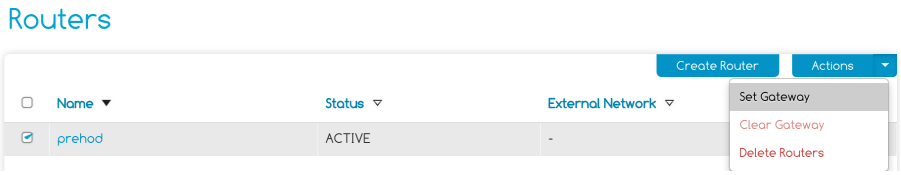
jo bomo povezali v novo omrežje in ji določili ustrezna varnostna pravila. Najprej običajno dodamo ključ SSH za dostop do instanc. V zavihku *Security* lahko pod *Keypairs* s klikom na *Import Keypair* enostavno uvozimo svoj javni ključ (slika 4.9). Pod *Security Groups* dodamo še varnostno skupino



Slika 4.9: Uvoz javnega ključa SSH za dostop do instanc.

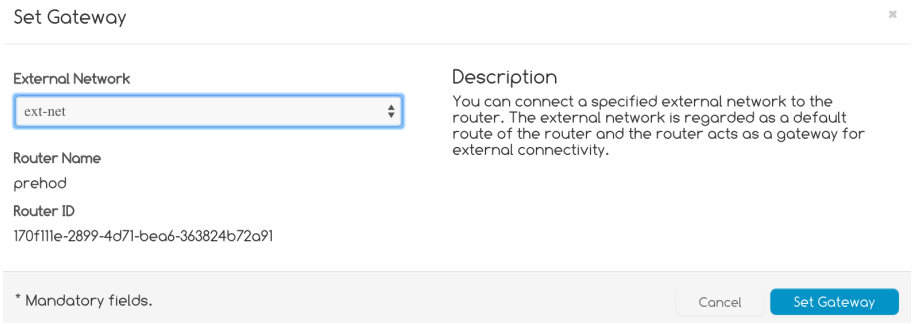
SSH, ki dovoljuje javni dostop SSH do vseh instanc, ki so v tej skupini. Ker je

okolje popolnoma prazno, dodamo usmerjevalnik, ki bo naše zasebno omrežje povezoval z zunanjim svetom. Med usmerjevalniki zgolj kliknemo na gumb

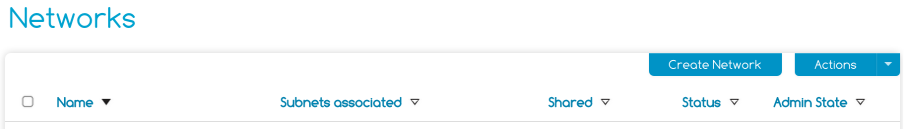


Slika 4.10: Določanje privzetega prehoda navideznemu usmerjevalniku.

Create Router, vnesemo želeno ime usmerjevalnika in potrdimo izbiro. S klikom na *Actions* in *Set gateway* (slika 4.10) usmerjevalniku določimo privzet prehod oziroma javno omrežje (slika 4.11). Sedaj dodamo še interno omrežje

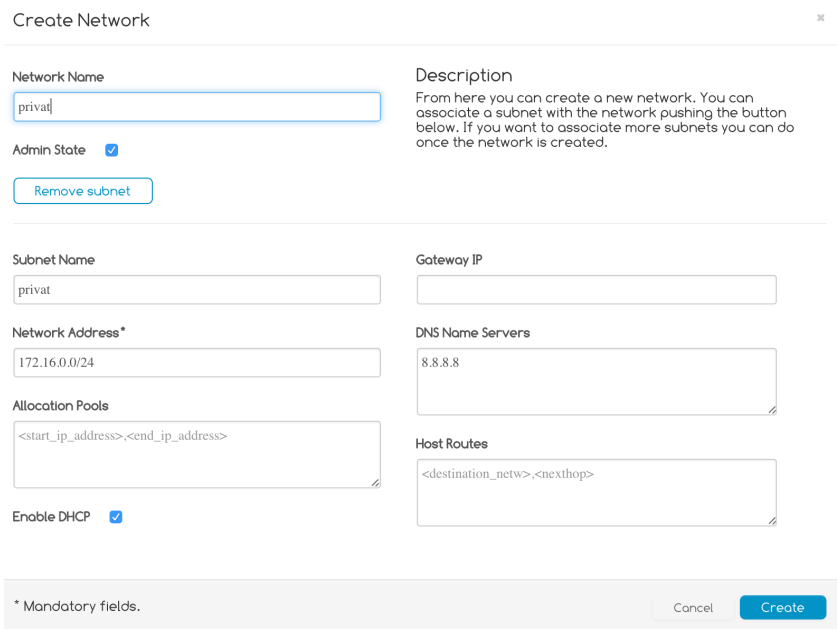


Slika 4.11: Izbira javnega omrežja.



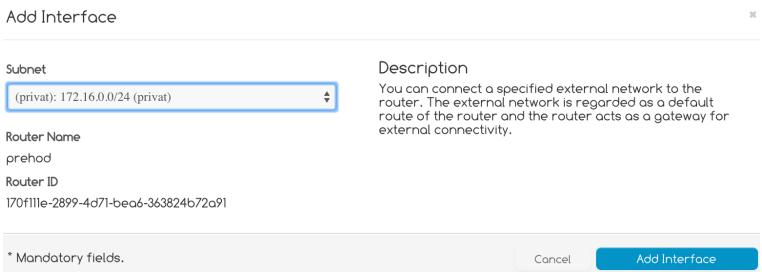
Slika 4.12: Interna omrežja.

s klikom na *Create network* na pogledu *Networks* (slika 4.12). Kot vidimo na sliki 4.13, lahko izbiramo ime omrežja in omrežnega segmenta, naslovni prostor, prisotnost ali odsotnost strežnika DHCP. Po potrebi lahko vnesemo



Slika 4.13: Dodajanje internega omrežja.

še naslov strežnikov DNS, ki jih strežnik DHCP posreduje instancam, in morebitne statične zapise v usmerjevalni tabeli. Interno omrežje povežemo z usmerjevalnikom tako, da usmerjevalniku dodamo vmesnik v tem omrežju. Poiščemo usmerjevalnik, ki smo ga ustvarili, in v zavihku *Interfaces* s klikom na *Add Interface* dodamo vmesnik, kot prikazuje slika 4.14. Izstavitev in-

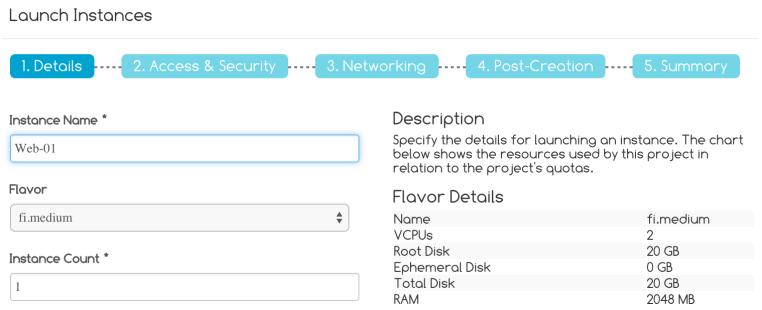


Slika 4.14: Dodajanje vmesnika na usmerjevalniku.

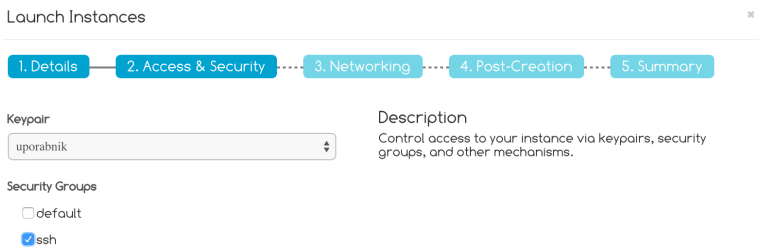
stanc je sedaj zelo preprosta. Med slikami izberemo želen operacijski sistem s klikom na *Launch* (slika 4.15). Vnesemo želeno ime instance (slika 4.16),

Slika 4.15: Izbira slike.

izberemo ključ SSH za dostop, varnostne skupine, katerim pripada instanca (slika 4.17) ter omrežje, v katero bo povezana.



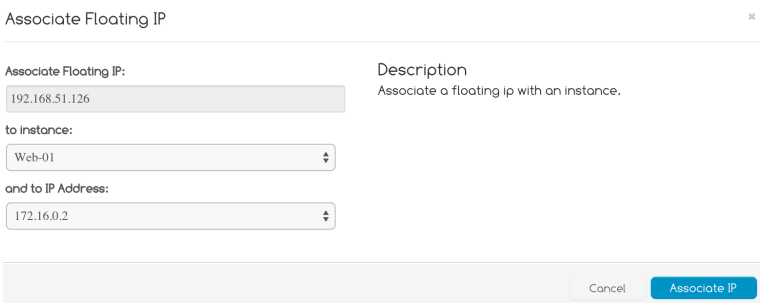
Slika 4.16: Izbira velikosti in imena instance.



Slika 4.17: Izbira ključa SSH in varnostnih skupin.

Zagon instance lahko spremljamo v zavihku log ali connection, kjer nam je na voljo konzolni dostop do instance. Ob zagonu orodje cloud-init poskrbi za avtomatsko prilagoditev instance na podlagi metapodatkov. Med drugim se v datoteko .ssh/authorized_keys doda javni ključ SSH, ki smo ga

izbrali pri izstavitvi instance. Dostop SSH s ključi je priporočen način upravljanja instanc, saj je dostop z geslom zaradi varnostnih razlogov običajno onemogočen. V kolikor zaganjamo instance z operacijskim sistemom Windows, kjer je oddaljen dostop možen preko RDP z uporabniškim imenom in geslom, se ob prvem zagonu generira naključno geslo za uporabnika Admin, ki se zakriptira z našim javnim ključem. Geslo lahko nato preberemo s pomočjo odjemalca in ga odkriptiramo s svojim zasebnim ključem. Za zdaj portal tega ne omogoča. Če nimamo neposrednega dostopa do programskega vmesnika, lahko uporabimo slike, kjer geslo nastavimo ob prvi prijavi preko kozole, ali pa je geslo prednastavljeno in ga po potrebi spremenimo.



Slika 4.18: Pripenjanje plavajočega naslova IP.

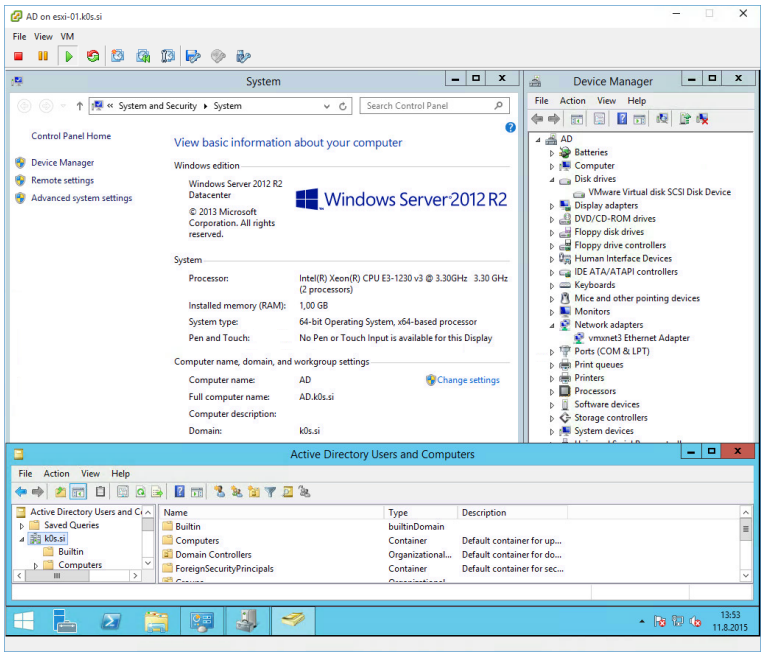
Instanci lahko dodelimo še plavajoči naslov IP za zunanji dostop. Zavihek *Floating IPs* najdemo pod *Security*, kjer z *Allocate IP to Project* pridobimo naslov iz bazena naslovov in ga nato pripnemo instanci s klikom na *Actions* ter *Associate IP*, pri čemer moramo izbrati instanco in interni naslov IP v katerega se preslika plavajoči naslov. Sedaj lahko do instance dostopamo enostavno s klientom SSH preko plavajočega naslova IP.

4.4.3 Migracija obstoječih strežnikov

Kadar si obstoječih aplikacij ne želimo postavljati na novo in te tečejo na virtualnih strojih s kompatibilnim operacijskim sistemom, jih lahko skupaj s celotnim sistemom relativno hitro in brez večjih sprememb preselimo v

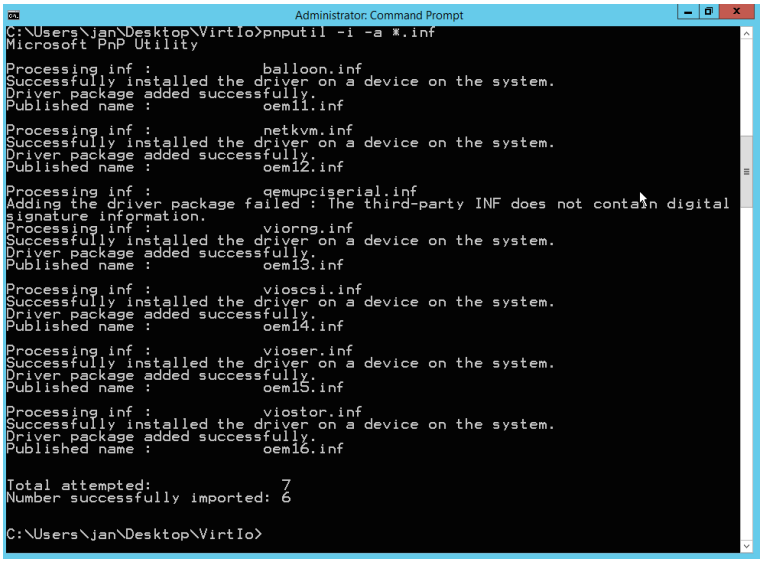
oblačno infrastrukturo. Za konec smo zato pregledali še možnosti za selitev obstoječih strežnikov v oblačno infrastrukturo na platformi FIWARE.

V osnovi moramo v obstoječi sistem namestiti ustrezne gonilnike, v našem primeru so to VirtIO, prenesti vse navidezne diske v bločno shrambo in zagnati instanco primernega tipa iz prenesenega sistemskega diska. Na žalost bločna shramba v tej fazi ne omogoča uvoza navideznih diskov, lahko pa navidezni disk najprej naložimo v shrambo slik in iz slike ustvarimo novo bločno enoto, iz katere se zažene instanca. Za primer smo uporabili obstoječi



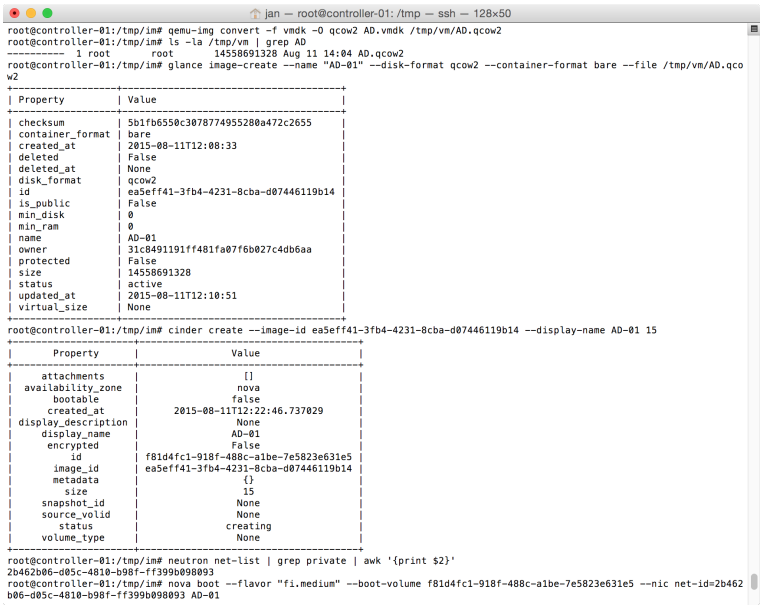
Slika 4.19: Obstoječi domenski strežnik Windows Server 2012 R2.

domenski strežnik Windows Server 2012 R2, ki teče na hipervizorju ESXi (slika 4.19). Pred migracijo smo z orodjem pnputil v sistem dodali gonilnike VirtIO (slika 4.20). Strežnik smo nato ugasnili in skopirali navidezni disk, ki je v formatu VMDK. Z orodjem qemu-img smo spremenili format navideznega diska v QCOW2 in disk uvozili v storitev shrambe slik. Samopostrežni portal uvoza slik končnim uporabnikom na žalost še ne omogoča. Ravno tako manjka možnost zagona instance iz bločne enote, ki predsta-

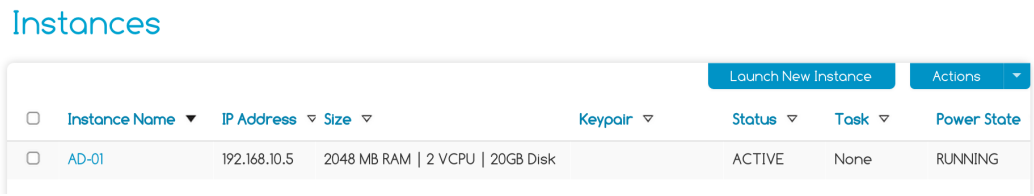


Slika 4.20: Dodajanje gonilnikov z orodjem pnputil.

vlja trajno hrambo, in jo v tem primeru potrebujemo tudi za sistemski disk. Kljub temu to lahko storimo direktno preko tekstovnega odjemalca oziroma programskega vmesnika. Postopek migracije in zagona instance lahko vidimo na sliki 4.21. Ugotovili smo, da se instanca zaradi napake pri dostopu do diska ne zažene. Dodajanje gonilnikov z orodjem pnputil v primeru krmilnika diska ne zadostuje. Našli smo rešitev za pravilno namestitev krmilnika diska z vmesnim korakom. Ročno smo ustvarili navidezni stroj na hipervizorju KVM, pri čemer smo sistemski disk dodali preko krmilnika IDE, ki deluje brez dodatnih gonilnikov, dodaten prazen disk pa preko krmilnika VirtIO, da se je ob zagonu krmilnik VirtIO pravilno namestil. Navidezni disk smo lahko nato ponovno pripeli oblačni instanci, ki se je tokrat zagnala brez težav. Preseljen strežnik je sedaj viden tudi v samopostrežnem portalu (slika 4.22). Kot vidimo na sliki 4.22, smo strežnik uspešno preselili na oblačno platformo. Preizkusili smo še storitve, ki tečejo na preseljenem strežniku. Zagnali smo instanco z operacijskim sistemom Windows 8.1 in jo uspešno priključili domeni na domenskem strežniku. Ugotovili smo, da preseljene storitve delujejo brez težav.

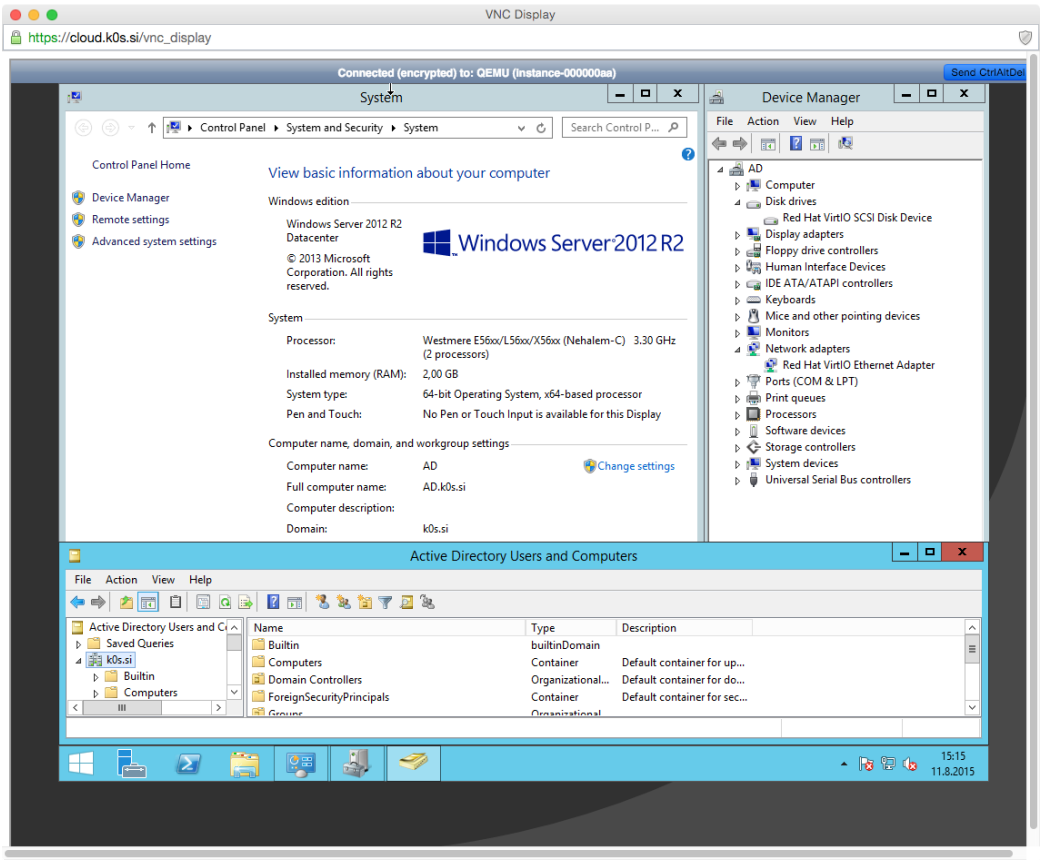


Slika 4.21: Postopek migracije navideznega diska in zagon instance.



Slika 4.22: Preseljen strežnik v samopostrežnem portalu.

Pri migraciji bi se radi izognili dvojnemu kopiranju diska najprej v shrambo slik in nato iz slike v bločno enoto, iz katere se zažene instanca. Poiskali smo hitrejši način migracije navideznega diska, ki pa uradno ni podprt. Navidezni disk smo tokrat z orodjem qemu-img spremenili v surovi format, ki ga uporablja bločna shramba. V bločni shrambi smo ustvarili prazno bločno enoto enake velikosti, kot je naš navidezni disk in navidezni disk skopirali direktno na sistem NAS na mesto prazne bločne enote. Gonilnike smo pripravili na enak način kot v prvem primeru in na koncu uspešno zagnali instanco iz bločne enote. Takšen način ni primeren za končne uporabnike oblačne infra-



Slika 4.23: Preseljen domenski strežnik.

strukture, vendar bi ga lahko implementirali v okviru bločne shrambe, kar bi poenostavilo migracijo obstoječih strežnikov.

Poglavje 5

Zaključek

V diplomskem delu smo definirali koncepte računalništva v oblaku, ki lahko nedvomno pripomore k večji učinkovitosti poslovnih procesov in zmanjševanju stroškov s pomočjo standardizacije, avtomatizacije in boljše izkoriščenosti virov. Pri tem smo spoznali različne postavitvene in storitvene modele oblaka ter izpostavili glavne značilnosti. Ogledali smo si tehnologije, ki te koncepte omogočajo, pri čemer smo se osredotočili na nudenje infrastrukture kot storitve. V nadaljevanju smo se podrobneje lotili odprte tehnološke platforme FIWARE, ki predstavlja alternativo uveljavljenim lastniškim rešitvam za postavitev oblačne infrastrukture. Glavni cilj je bila vzpostavitev in evalvacija oblaka na odprti platformi z namenom nudenja infrastrukture kot storitve.

Podrobno smo spoznali in preizkusili komponente, ki sestavljajo tehnološko platformo FIWARE, kar nam je omogočilo uspešno vzpostavitev oblačne infrastrukture. Preizkusili smo jo tako z uporabniškega kot administratorskega vidika ter analizirali njeno delovanje. Ugotovili smo, da rešitev zagotavlja bistvene značilnosti računalništva v oblaku in rešuje predvsem problem zaklepanja uporabnikov s široko podporo spodaj ležečih tehnologij in odprtimi aplikacijskimi programskimi vmesniki za enostavno integracijo. Kot glavni problem lahko izpostavimo odsotnost stabilnih izdaj, saj je večina komponent v trenutni fazi na voljo zgolj v obliki izvorne kode, navodila za namestitev pa so namenjena predvsem vzpostavitvi razvojnega okolja. Ugo-

tovili smo, da je postavitev infrastrukture kompleksen proces, ki zahteva zelo dobro načrtovanje. Dodatno kompleksnost predstavlja dejstvo, da je rešitev sestavljena iz kopice odprtokodne programske opreme, ki jo moramo ravno tako zelo dobro poznati. Menimo, da je postavitev v lastni režiji smiselna predvsem za večja okolja, saj zahteva precej optimizacij za zagotovitev optimalnega delovanja. Predvidevamo, da se to lahko spremeni z izdajo orodja ITBox, ki se ravno tako razvija v okviru projekta FIWARE. Omogočal naj bi namreč popolnoma avtomatsko vzpostavitev oblačne infrastrukture, vendar za zdaj še ne vsebuje vseh potrebnih komponent. Platforma FIWARE je sicer zanimiva predvsem za gostovanje sodobnih večslojnih oblačnih aplikacij, ki zagotavljajo visoko razpoložljivost na nivoju same aplikacije, in nekoliko manj za klasične aplikacije. Med postavitvijo smo namreč ugotovili, da so bile razširitve, ki bi omogočile poganjanje širšega nabora bremen z zagotavljanjem visoke razpoložljivosti na nivoju instance, zaenkrat v okviru odprtokodnega projekta opuščene.

Literatura

[1] M. B. Jurič, R. Dukarić in R. Povše, "Računalništvo v oblaku." [ONLINE]. Dosegljivo: http://www.soa.si/wp-content/uploads/2011/11/Delavnica-racunalni%C5%A1tvo-v-oblaku_zaUdelezence.pdf, 2015. [Dostopano 17. 6. 2015].

[2] Nutanix. [ONLINE]. Dosegljivo: http://www.nutanix.com/, 2015. [Dostopano 20. 6. 2015].

[3] P. Sharma, "Difference between isl and 802.1q." [ONLINE]. Dosegljivo: https://ciscohite.wordpress.com/2013/05/14/difference-between-isl-802-1q/, 2013. [Dostopano 20. 6. 2015].

[4] OpenStack, "Installation guide for ubuntu 14.04." [ONLINE]. Dosegljivo: http://docs.openstack.org/juno/install-guide/install/apt/content/, 2015. [Dostopano 23. 3. 2015].

[5] Rackspace, "Laying cinder block (volumes) in openstack, part 1." [ONLINE]. Dosegljivo: http://www.rackspace.com/blog/laying-cinder-block-volumes-in-openstack-part-1-the-basics/, 2014. [Dostopano 26. 7. 2015].

[6] K. Mestery, "Neutron modular layer 2 (ml2) overview." [ONLINE]. Dosegljivo: https://wiki.opendaylight.org/images/e/e9/ML2_Overview.pptx, 2015. [Dostopano 29. 7. 2015].

[7] R. Dua, "Openvswitch deep dive." [ONLINE]. Dosegljivo: http://www.slideshare.net/rajdeep/openvswitch-deep-dive?next_slideshow=1, 2013. [Dostopano 29. 7. 2015].

[8] T. Costello in B. Prohaska, "2013 trends and strategies," *IT Professional*, vol. 15, no. 1, str. 64–64, 2013.

[9] P. Mell in T. Grance, "The nist definition of cloud computing." [ONLINE]. Dosegljivo: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2011. [Dostopano 16. 6. 2015].

[10] L. Kaufman, "Data security in the world of cloud computing," *Security Privacy, IEEE*, vol. 7, str. 61–64, July 2009.

[11] R. Dukarić in M. B. Jurič, "Migracija obstoječih aplikacij na platforme za računalništvo v oblaku," *Uporabna informatika*, vol. 19, no. 3, str. 136–146, 2011.

[12] MJU, "Državni računalniški oblak." [ONLINE]. Dosegljivo: http://www.mju.gov.si/si/delovna_podrocja/informatika/drzavni_racunalniski_oblak/, 2015. [Dostopano 17. 6. 2015].

[13] R. Turnšek, "Pot v velike svetovne javne oblake." [ONLINE]. Dosegljivo: https://www.linkedin.com/pulse/pot-v-velike-svetovne-javne-oblake-robert-turn%C5%A1ek, 2015. [Dostopano 17. 6. 2015].

[14] M. Humphrey, Z. Hill, K. Jackson, C. van Ingen in Y. Ryu, "Assessing the value of cloudbursting: A case study of satellite image processing on windows azure," v *zborniku E-Science (e-Science), 2011 IEEE 7th International Conference on E-Science*, str. 126–133, Dec 2011.

[15] R. Moreno-Vozmediano, R. Montero in I. Llorente, "Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures," *Computer*, vol. 45, str. 65–72, Dec 2012.

[16] Amazon, "Elastic compute cloud." [ONLINE]. Dosegljivo: `http://aws.amazon.com/ec2/`, 2015. [Dostopano 18. 6. 2015].

[17] Google, "Compute engine." [ONLINE]. Dosegljivo: `https://cloud.google.com/compute/`, 2015. [Dostopano 18. 6. 2015].

[18] Rackspace, "Rackspace cloud servers." [ONLINE]. Dosegljivo: `http://www.rackspace.com/cloud/servers`, 2015. [Dostopano 18. 6. 2015].

[19] DigitalOcean. [ONLINE]. Dosegljivo: `https://www.digitalocean.com`, 2015. [Dostopano 18. 6. 2015].

[20] Microsoft, "Azure." [ONLINE]. Dosegljivo: `http://azure.microsoft.com/`, 2015. [Dostopano 18. 6. 2015].

[21] Softlayer, "Virtual servers." [ONLINE]. Dosegljivo: `http://www.softlayer.com/virtual-servers`, 2015. [Dostopano 18. 6. 2015].

[22] NIL, "Podatkovni centri." [ONLINE]. Dosegljivo: `http://www.nil.com/sl/resitve/podatkovno-sredisce/`, 2015. [Dostopano 18. 6. 2015].

[23] Optimus IT, "mojoblak.si." [ONLINE]. Dosegljivo: `http://mojoblak.si/oblak/`, 2015. [Dostopano 18. 6. 2015].

[24] Pošta Slovenije, "Posita." [ONLINE]. Dosegljivo: `https://www.posita.si/portal/sec/portal/default/Products?resetDefaultView=true&prodCategory=default`, 2015. [Dostopano 18. 6. 2015].

[25] Google, "App engine." [ONLINE]. Dosegljivo: `https://cloud.google.com/appengine/`, 2015. [Dostopano 18. 6. 2015].

[26] RadHat, "Openshift." [ONLINE]. Dosegljivo: `https://www.google.com/`, 2015. [Dostopano 18. 6. 2015].

[27] Heroku. [ONLINE]. Dosegljivo: http://heroku.com, 2015. [Dostopano 18. 6. 2015].

[28] FIWARE, "Pegasus paas manager." [ONLINE]. Dosegljivo: http://catalogue.fiware.org/enablers/paas-manager-pegasus, 2015. [Dostopano 18. 6. 2015].

[29] U. Sedlar, J. Bešter in A. Kos, "Računalništvo v oblaku v telekomunikacijah in primeri uporabe." [ONLINE]. Dosegljivo: http://www.ltfe.org/wp-content/uploads/2011/11/2-Urban-Sedlar-Janez-Bester-Andrej-Kos-VITELnov2011.pdf, 2011. [Dostopano 18. 6. 2015].

[30] IBM, "Virtualization ibm." [ONLINE]. Dosegljivo: https://www.ibm.com/developerworks/community/blogs/ibmvirtualization/entry/kvm_myths_uncovering_the_truth_about_the_open_source_hypervisor?lang=en, 2012. [Dostopano 19. 6. 2015].

[31] Canonical, "Linux containers." [ONLINE]. Dosegljivo: https://linuxcontainers.org, 2015. [Dostopano 19. 6. 2015].

[32] Microsoft, "New windows server containers and azure support for docker." [ONLINE]. Dosegljivo: http://azure.microsoft.com/blog/2014/10/15/new-windows-server-containers-and-azure-support-for-docker/, 2015. [Dostopano 19. 6. 2015].

[33] Arstechnica, "The great disk drive in the sky: How web giants store big—and we mean big—data." [ONLINE]. Dosegljivo: http://arstechnica.com/?p=37620, 2015. [Dostopano 19. 6. 2015].

[34] EMC, "Scaleio." [ONLINE]. Dosegljivo: http://www.emc.com/storage/scaleio/index.htm, 2015. [Dostopano 21. 6. 2015].

[35] VMware, "vsan." [ONLINE]. Dosegljivo: http://www.vmware.com/products/virtual-san, 2015. [Dostopano 21. 6. 2015].

[36] FIWARE, "Iaas dcrm." [ONLINE]. Dosegljivo: `https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/IaaS_Data_Center_Resource_Management_-_Installation_and_Administration_Guide`, 2014. [Dostopano 24. 3. 2015].

[37] ETSI, "Network functions virtualisation." [ONLINE]. Dosegljivo: `http://www.etsi.org/technologies-clusters/technologies/nfv`, 2015. [Dostopano 25. 6. 2015].

[38] PaloAlto, "Virtualized firewalls." [ONLINE]. Dosegljivo: `https://www.paloaltonetworks.com/products/platforms/virtualized-firewalls/vm-series/overview.html`, 2015. [Dostopano 25. 6. 2015].

[39] FI-PPP, "The fi ppp programme." [ONLINE]. Dosegljivo: `http://www.fi-ppp.eu/about/`, 2015. [Dostopano 12. 4. 2015].

[40] FIWARE, "About us." [ONLINE]. Dosegljivo: `http://www.fiware.org/about-us/`, 2015. [Dostopano 12. 4. 2015].

[41] FIWARE, "Lab." [ONLINE]. Dosegljivo: `http://lab.fiware.org`, 2015. [Dostopano 13. 4. 2015].

[42] XIFI, "Fiware lab federation members." [ONLINE]. Dosegljivo: `https://fi-xifi.eu/about-xifi/federation-members.html`, 2014. [Dostopano 8. 6. 2015].

[43] FIWARE, "Fiware ops." [ONLINE]. Dosegljivo: `http://www.fiware.org/fiware-operations/`, 2015. [Dostopano 15. 4. 2015].

[44] MIZS, "Program pospeševalcev fiware." [ONLINE]. Dosegljivo: `http://www.mizs.gov.si/si/obzorje2020/razpisi/program_pospesevalcev_fiware_razpisi_za_projekte_za_majhna_in_srednja_podjetja/`, 2015. [Dostopano 15. 4. 2015].

[45] FIWARE, "Fiware mundus." [ONLINE]. Dosegljivo: http://www.fiware.org/mundus/, 2015. [Dostopano 14. 6. 2015].

[46] FIWARE, "Frequently asked questions." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_FAQ, 2014. [Dostopano 12. 4. 2015].

[47] FIWARE, "Overall fiware vision." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Overall_FI-WARE_Vision, 2011. [Dostopano 15. 2. 2015].

[48] F. López, "Introduction to fiware open ecosystem." [ONLINE]. Dosegljivo: http://www.slideshare.net/flopezaguilar/introduction-to-fiware-open-ecosystem, 2014. [Dostopano 12. 6. 2015].

[49] F. Consortium, "Fi-ware high-level description." [ONLINE]. Dosegljivo: http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fi-ware/fi-ware-d222.pdf-FI-WAREProductVisionfrontpage, 2011. [Dostopano 15. 2. 2015].

[50] FIWARE, "Cloud hosting architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Cloud_Hosting_Architecture, 2015. [Dostopano 6. 6. 2015].

[51] FIWARE, "Data/context management architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Data/Context_Management_Architecture, 2015. [Dostopano 6. 6. 2015].

[52] FIWARE, "Architecture of applications, services and data delivery." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Architecture_of_

Applications_and_Services_Ecosystem_and_Delivery_Framework, 2015. [Dostopano 6. 6. 2015].

[53] FIWARE, "Internet of things (iot) services enablement architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Internet_of_Things_(IoT)_Services_Enablement_Architecture, 2015. [Dostopano 6. 6. 2015].

[54] FIWARE, "Interface to networks and devices (i2nd) architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Interface_to_Networks_and_Devices_(I2ND)_Architecture, 2015. [Dostopano 6. 6. 2015].

[55] FIWARE, "Security architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Security_Architecture, 2015. [Dostopano 6. 6. 2015].

[56] FIWARE, "Advanced web ui architecture." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Advanced_Web_UI_Architecture, 2015. [Dostopano 6. 6. 2015].

[57] FIWARE, "Fiware open specifications." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Summary_of_FI-WARE_Open_Specifications, 2015. [Dostopano 6. 6. 2015].

[58] FIWARE, "Fiware forge tracker." [ONLINE]. Dosegljivo: https://forge.fiware.org/tracker/?atid=163&group_id=7&func=browse, 2015. [Dostopano 13. 6. 2015].

[59] FIWARE, "Materializing cloud hosting in fi-ware." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Materializing_Cloud_Hosting_in_FI-WARE, 2015. [Dostopano 14. 6. 2015].

[60] FIWARE, "Fiware open specifications." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Technical_Roadmap, 2015. [Dostopano 7. 6. 2015].

[61] FIWARE, "Fiware agile development methodology." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Agile_Development_Methodology, 2015. [Dostopano 13. 6. 2015].

[62] FIWARE, "Quick fiware tour guide for developers." [ONLINE]. Dosegljivo: http://www.fiware.org/tour-guide/, 2015. [Dostopano 14. 6. 2015].

[63] FIWARE, "Working with the fiware catalogue." [ONLINE]. Dosegljivo: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Working_with_the_FIWARE_catalogue, 2015. [Dostopano 14. 6. 2015].

[64] R. Boden, "Openstack keystone workflow and token scoping." [ONLINE]. Dosegljivo: http://bodenr.blogspot.com/2014/03/openstack-keystone-workflow-token.html, 2014. [Dostopano 23. 7. 2015].

[65] FIWARE, "Dcrm open specification." [ONLINE]. Dosegljivo: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.DCRM, 2013. [Dostopano 24. 3. 2015].

[66] IBM, "Ibm cloud manager with openstack." [ONLINE]. Dosegljivo: http://www.ibm.com/developerworks/servicemanagement/cvm/sce/, 2015. [Dostopano 16. 4. 2015].

[67] OpenStack, "Configuration reference." [ONLINE]. Dosegljivo: `http://docs.openstack.org/juno/config-reference/content/`, 2015. [Dostopano 23. 3. 2015].

[68] OpenStack, "Contributor documentation." [ONLINE]. Dosegljivo: `http://docs.openstack.org/developer/openstack-projects.html`, 2015. [Dostopano 24. 7. 2015].

[69] R. Bryant, "A new nova service: nova-conductor." [ONLINE]. Dosegljivo: `http://blog.russellbryant.net/2012/11/19/a-new-nova-service-nova-conductor/`, 2012. [Dostopano 25. 7. 2015].

[70] Ceph, "Block devices and openstack." [ONLINE]. Dosegljivo: `http://ceph.com/docs/master/rbd/rbd-openstack/`, 2014. [Dostopano 26. 7. 2015].

[71] Rackspace, "Laying cinder block (volumes) in openstack, part 2." [ONLINE]. Dosegljivo: `http://www.rackspace.com/blog/laying-cinder-block-volumes-in-openstack-part-2`, 2014. [Dostopano 26. 7. 2015].

[72] R. Boswell, "What commands are called during the startup of the neutron-plugin-openvswitch-agent?." [ONLINE]. Dosegljivo: `http://www.revolutionlabs.net/2013/11/what-commands-are-called-during-startup_28.html`, 2013. [Dostopano 30. 7. 2015].

[73] B. Derzhavets, "How vms access metadata via qrouter-namespace in juno." [ONLINE]. Dosegljivo: `http://bderzhavets.blogspot.com/2014/11/access-to-metadata-via-qrouter.html`, 2014. [Dostopano 27. 7. 2015].

[74] M. Rojas, "There's real magic behind openstack neutron." [ONLINE]. Dosegljivo: `http://pinrojas.com/2014/07/29/`

theres-real-magic-behind-openstack-neutron/, 2014. [Dostopano 2. 6. 2015].

[75] RadHat, "Networking in too much detail." [ONLINE]. Dosegljivo: https://www.rdoproject.org/Networking_in_too_much_detail, 2014. [Dostopano 2. 6. 2015].

[76] OpenStack, "Cloud administrator guide." [ONLINE]. Dosegljivo: http://docs.openstack.org/admin-guide-cloud/content/ch_networking.html, 2015. [Dostopano 23. 3. 2015].

[77] FIWARE, "Self service interfaces open specification." [ONLINE]. Dosegljivo: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.SelfServiceInterfaces, 2015. [Dostopano 1. 8. 2015].

[78] FIWARE, "Self service interfaces - user guide." [ONLINE]. Dosegljivo: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Self_Service_Interfaces_-_User_Guide, 2015. [Dostopano 1. 8. 2015].

[79] ForeverJS, "Forever." [ONLINE]. Dosegljivo: https://github.com/foreverjs/forever, 2015. [Dostopano 1. 8. 2015].

[80] RadHat, "Cloud-init." [ONLINE]. Dosegljivo: https://cloudinit.readthedocs.org/en/latest/, 2015. [Dostopano 25. 7. 2015].